## DOCTOR OF PHILOSOPHY

# Markerless multiple-view human motion analysis using swarm optimisation and subspace learning

John, Vijay

*Award date:*
2011

DOCTOR OF PHILOSOPHY

# Markerless multiple-view human motion analysis using swarm optimisation and subspace learning

Vijay John

2011

University of Dundee

# Markerless Multiple-View Human Motion Analysis using Swarm Optimisation and Subspace Learning

Vijay John

School of Computing

University of Dundee

May 2011

# Contents

# List of Figures

# List of Tables

# Acknowledgements

First, I would like to thank my supervisor, Manuel, for his continued support and encouragement throughout my research. I appreciate him for giving me the motivation to achieve more and setting the bar high. I really enjoyed the weekly supervisory meetings, and it was a pleasure doing my research under him. I take this opportunity to express my gratitude to him for being an understanding and patient supervisor, and a wonderful person, while not compromising on the quality of research.

I also wish to acknowledge and thank Spela for the innumerable discussions, talks and interactions, which went a long way in helping me to ease into the research environment. I also would like to thank her for meticulously proof-reading a big portion of my work.

I also would like to thank the Prof. Adrian Hilton, for the numerous stimulating discussions we have had. A special mention is needed for Dr. Simon Prince for providing valuable insight into dimensionality reduction. Furthermore, I would like to thank Prof. Stephen McKenna for being kind enough, to spare some of his time to explain and approve some of the concepts. I would like to thank Prof. Rami Abboud for inviting us to visit the Institute of Motion Analysis and Research at the Ninewells hospital. Finally, I would like to express my gratitude to Prof. Mel Seigel for instilling in me the desire to be inquisitive, during my Master.

A pleasant working environment went a long way in enabling me to complete this

thesis, and I am grateful to the vision group in Dundee for making it a wonderful environment to research, especially Gregor, Jerry, Khai Sing, Adria, Telmo, and Roy.

I would also like to acknowledge the EPSRC, Dundee Discovery Scholarship, and the School of Computing, for providing the studentship, scholarship, and travel funds which enabled me to pursue my PhD and attend conferences to present my work.

On the personal front, I really can't thank my parents enough (Vasanthi and John), for without their painstaking sacrifice of their health and personal happiness, none of this would be possible. I am always grateful for what they have done for me, which neither words or deeds can explain. I also would like to thank my grandpa, Solomon for having a burning desire to see me pursue a doctorate. I also would like to thank my brothers, Arul, Ajay, Mani, and Spurgeon, for keeping me connected to the outside world during my research. A heart-felt gratitude to Arul for listening to my daily whining and rants, and helping me through the difficult times. My aunt, also deserves a special mention for teaching me to never give up, and persevere, irrespective of life's hardships. A big thank you is due to all of my well-wishers, relatives, especially Rev. Sunderasekaran and Shanti, for treating me like their son. Special mention to the friends, who wished me well and who prayed for me, especially Mrs. Suganthi and Mrs. Kasturi. Last, but not the least, I would like to thank, my loving fiancee, Reena, for coming into my life at the right time, and being supportive and understanding, considering my limited availability to spend time with her.

Above all, I would like to thank God, for being the source of my strength, my all in all, and without whose grace, this would not have been possible.

*Soli Deo Gloria.*

# Associated publications

This work has been reported in research papers accepted by a number of events and journal, namely: the International Conference on Computer Vision Theory and Applications (VISAPP) in 2009 (oral presentation); the Communications in Computer and Information Science in 2010 (invited paper); Image and Vision Computing Journal in 2010 (journal); Applications of Evolutionary Computation (EvoAPPS) in 2010 (oral presentation); the British Machine Vision Conference Postgraduate Workshop in 2010 (oral presentation); and lastly, at the ACM Multimedia International workshop on 3D Video Processing, (3DVP) in 2010 (oral presentation). These publications are listed below.

Additional dissemination material included a poster presentation at the attended BMVA Computer Vision Summer School in 2008 and SICSA Demofest at Edinburgh in 2009. The PhD research project was selected for publication in the editorial, 'Science Scotland', published by the Royal Society of Edinburgh.

V. John, E. Trucco. Multiple View Human Articulated Tracking using Charting and Particle Swarm Optimisation, *In Proc. ACM Multimedia International workshop on 3D Video Processing*, 3DVP 2010, 2010

V. John, E. Trucco, S. J. McKenna, Markerless Human Motion Capture using Charting and Manifold Constrained Particle Swarm Optimisation, *In Proc. BMVC UK postgrad. workshop*, 2010. [**Best Paper Prize**]

V. John, S. Ivekovic, E. Trucco, Markerless Human Motion Capture Using

Hierarchical Particle Swarm Optimisation, *Chapter in Computer Vision, Imaging and Computer Graphics. Theory and Applications*, volume 68, 2010. [Invited Paper]

V. John, E. Trucco, S. Ivekovic, Markerless human articulated tracking using hierarchical particle swarm optimisation, *In Image Vision Computing.*, volume 28, 2010.

S. Ivekovic, V. John, E. Trucco, Markerless Multi-view Articulated Pose Estimation Using Adaptive Hierarchical Particle Swarm Optimisation, *In EvoApplications* 2010.

V. John, S. Ivekovic, E. Trucco, Articulated Human Motion Tracking with HPSO, *In VISSAPP*, 2009.

# Declaration by the author

I hereby declare that I am the author of this thesis; that all references cited have been consulted by me; that the work of which this thesis is a record has been done by me, and that it has not been previously accepted for a higher degree.

Signed

Vijay John

May 2011

# Declaration by the supervisor

I hereby certify that Mr Vijay John has satisfied all the terms and conditions of the regulations made under Ordinances 12 and 39 and has completed the required nine terms of research to qualify in submitting this thesis in application for the degree of Doctor of Philosophy.

Signed

Prof. Emanuele Trucco

May 2011

**Abstract**

The fundamental task in human motion analysis is the extraction or capture of human motion and the established industrial technique is marker-based human motion capture. However, marker-based systems, apart from being expensive, are obtrusive and require a complex, time-consuming experimental setup, resulting in increased user discomfort. As an alternative solution, research on markerless human motion analysis has increased in prominence. In this thesis, we present three human motion analysis algorithms performing markerless tracking and classification from multiple-view studio-based video sequences using particle swarm optimisation and charting, a subspace learning technique.

In our first framework, we formulate, and perform, human motion tracking as a multi-dimensional non-linear optimisation problem, solved using particle swarm optimisation (PSO), a swarm-intelligence algorithm. PSO initialises automatically, does not need a sequence-specific motion model, functioning as a black-box system, and recovers from temporary tracking divergence through the use of a powerful hierarchical search algorithm (HPSO). We compare experimentally HPSO with particle filter, annealed particle filter and partitioned sampling annealed particle filter, and report similar or better tracking performance. Additionally we report an extensive experimental study of HPSO over ranges of values of its parameters and propose an automatic-adaptive extension of HPSO called as adaptive particle swarm optimisation.

Next, in line with recent interest in subspace tracking, where low-dimensional subspaces are learnt from motion models of actions, we perform tracking in a low-dimensional subspace obtained by learning motion models of common actions using charting, a nonlinear dimensionality reduction tool. Tracking takes place in the subspace using an efficient modified version of particle swarm optimisation. Moreover, we perform a fast and efficient pose evaluation by representing the observed image data, multi-view silhouettes, using vector-quantized shape contexts and learning the mapping from the action subspace to shape space using multi-variate relevance vector machines. Tracking results with various action sequences demonstrate the good accuracy and performance of our approach.

Finally, we propose a human motion classification algorithm, using charting-based low-dimensional subspaces, to classify human action sub-sequences of varying lengths, or snippets of poses. Each query action is mapped to a single subspace space, learnt from multiple actions. Furthermore we present a system in which, instead of mapping multiple actions to a single subspace, each action is mapped separately to its action-specific subspace. We adopt a multi-layered subspace classification scheme with layered pruning and search. One of the search layers involves comparing the input snippet with a sequence of key-poses extracted from the subspace. Finally, we identify the minimum length of action snippet, of skeletal features, required for accurate classification, using competing classification systems as the baseline. We test our classification component on HumanEva and CMU mocap datasets, achieving similar or better classification accuracy than various comparable systems.

# List of symbols

An effort was made to avoid ambiguities in the notation used in mathematical descriptions. Lowercase Roman letters denote scalar variables, as in $x$, except when denoting functions. Lowercase bold Roman letters denote vectors, as in $\mathbf{x}$, and uppercase Roman letters denote constants, as in $N$. Uppercase bold letters denote matrices, as in $\mathbf{Y}$. Whenever possible, we represent the set of vectors using uppercase Roman letters, and with the same letter as the vectors, for example, set of vectors $\mathbf{x}$ is represented as $\mathbf{X}$.

The following Tables list the symbols that are shared across the discussions of different techniques, as well as those that are used in the description of specific techniques.

| Chapter 3. Hierarchical Particle Swarm Optimisation(PSO) | |
|---|---|
| $N$ | Number of particles |
| $\mathbf{X}_t$ | State, or particle set, at time $t$ (Particle filter) |
| $\mathbf{y}_t$ | Observation at time $t$ (Particle filter) |
| $\mathbf{x}_t^i$, $i \in \{1, \ldots, N\}$ | $i-th$ particle at time $t$ (Particle filter) |
| $\pi_t^i$, $i \in \{1, \ldots, N\}$ | Normalised weight of $i-th$ particle at time $t$ (Particle filter) |
| $\beta$ | Smoothing parameter (Annealed Particle Filter) |
| $d$ | Dimension of search space (PSO) |
| $\mathbf{a}$ | Search constraint vector (PSO) |
| $\mathbf{b}$ | Search constraint vector (PSO) |
| $\mathbf{V}$ | Set of velocity vectors (PSO) |
| $\mathbf{v}^i$, $i \in \{1, \ldots, N\}$ | $i-th$ velocity vector (PSO) |
| $\mathbf{P}$ | Set of personal best vectors (PSO) |
| $\mathbf{p}^i$, $i \in \{1, \ldots, N\}$ | $i-th$ personal best vector (PSO) |
| $g$ | Index of global best particle in swarm (PSO) |
| $\omega$ | Inertia parameter (PSO) |
| $\phi_1$ | Social component (PSO) |
| $\phi_2$ | Cognition component (PSO) |
| $A$ | Starting value of inertia (PSO) |
| $C$ | Number of PSO iterations |
| $K$ | Number of joints in body |
| $r$ | Co-ordinates of root in human body pose |
| $\alpha, \beta, \gamma$ | Rotational degrees of freedom of joints in human body pose |
| $\Sigma^e(\mathbf{x},\mathbf{z})$ | Edge-based cost function with state $\mathbf{x}$ and image $\mathbf{z}$ |
| $\Sigma^s(\mathbf{x},\mathbf{z})$ | Silhouette-based cost function with state $\mathbf{x}$ and image $\mathbf{z}$ |
| $\mathbf{x}_s^e$ | Pose estimated at $s-th$ hierarchical step (A-PSO) |
| $\tau_0, \tau_1$ | Cost function thresholds (A-PSO) |

| Chapter 4. Charting-based Subspace Tracking | |
|---|---|
| $D$ | Dimension of body joint angles |
| $d$ | Dimension of joint angles subspace |
| $\mathbf{Y}$ | Sequence of $D$-dimensional joint angles |
| $\mathbf{X}$ | Sequence of $d$-dimensional subspace representation |
| $\mathbf{y}^i$, $i \in \{1, \ldots, N\}$ | *i-th* joint angle feature vector |
| $\mathbf{x}^i$, $i \in \{1, \ldots, N\}$ | *i-th* subspace feature vector |
| $f(\mathbf{y})$ | Forward mapping function (Subspace learning) |
| g($\mathbf{x}$) | Inverse mapping function (Subspace learning) |
| $\mathbf{W}$ | Mapping matrix (Subspace learning) |
| $\epsilon$ | Gaussian noise (Subspace learning) |
| $N$ | Number of frames in video sequence |
| $r$ | Local linear scale (Charting) |
| $c(r)$ | Growth rate(Charting) |
| $\mu$ | Mean of Gaussian Mixture Model (Charting) |
| $\Sigma$ | Covariance of Gaussian Mixture Model (Charting) |
| $m(\mu)$ | Measure of co-locality (Charting) |
| $\mathbf{U}$ | Locally linear subspace representation (Charting) |
| $\mathbf{u}^i$, $i \in \{1, \ldots, N\}$ | *i-th* locally linear subspace feature vector(Charting) |
| $\mathbf{G}$ | Affine transform (Charting) |
| $\mathbf{F}$ | Indicator matrix (Charting) |
| $d$,$e$ | Cartesian pixel co-ordinates (Fourier descriptors) |
| $z$ | Complex pixel co-ordinates (Fourier descriptors) |
| $f$ | Fourier co-efficients (Fourier descriptors) |
| $\mathbf{V}$ | Input-output training pair of vectors (Regression) |
| $\mathbf{v}^i$, $i \in \{1, \ldots, N\}$ | *i-th* input-output training pair of vectors (Regression) |
| $\mathbf{R}$ | Input training vectors (Regression) |
| $\mathbf{Z}$ | Output training vectors (Regression) |
| $\phi(r)$ | Set of basis functions (Regression) |
| $\mathbf{C}$ | Weight matrix(Regression) |
| $\mathbf{S}$ | Noise matrix(Regression) |
| $\alpha$ | Hyperparameter of basis function(Regression) |
| $\Phi$ | Design matrix (Regression) |

| Chapter 5. Charting-based Subspace Multi-layered Classification | |
|---|---|
| $D$ | Dimension of body joint angles |
| $d$ | Dimension of joint angles subspace |
| $\mathbf{Y}$ | Sequence of $D$-dimensional joint angles |
| $\mathbf{X}$ | Sequence of $d$-dimensional subspace representation |
| $\mathbf{v}_t^c$ | Subspace feature vector in *c-th* partition |
| $\mathbf{x}^c$ | Point co-ordinates in $\mathbf{v}_t^c$ in *c-th* partition |
| $\mathbf{s}^c$ | Spacing co-ordinates in $\mathbf{v}_t^c$ in *c-th* partition |
| $\mathbf{W}^c$ | Set of cluster centers of $\mathbf{v}_t^c$ in *c-th* partition |
| $\mathbf{v}_f^c$ | Key-frame subspace feature vector in *c-th* partition |
| $\mathbf{x}_f^c$ | Point co-ordinates in $\mathbf{v}_f^c$ in *c-th* partition |
| $\mathbf{s}_f^c$ | Spacing co-ordinates in $\mathbf{v}_f^c$ in *c-th* partition |
| $\mathbf{L}^c$ | Set of cluster centers of $\mathbf{v}_f^c$ in *c-th* partition |
| $\mathbf{A}$, $\mathbf{B}$ | Sequence of feature vectors (Distance measures) |
| $\mathbf{C}$ | Local distance matrix (Dynamic time warping) |
| $\mathbf{D}$ | Accumulated distance matrix (Dynamic time warping) |
| $\upsilon$ | Candidate action labels (Classification framework) |
| $\mathbf{Y}^q$ | Query snippet |
| $\eta$ | Point-to-set Hausdorff distance value |
| $\mathbf{U}^c$ | Set of nearest cluster subspace feature vectors in *c-th* partition |
| $\mathbf{P}^c$ | Final set of candidate subspace feature vectors in *c-th* partition |

# Chapter 1

# Introduction

This thesis reports work carried out by the author at the School of Computing of the University of Dundee concerning the markerless human motion tracking and classification of multiple-view video sequences using particle swarm optimisation and charting. This work started in September, 2007.

Human motion analysis is an important problem tackled by the computer-vision research community for a long time. The fundamental problem is the extraction of relevant human motion information, such as sequences of 3D joint angles, (body positions) or as single objects from video sequences, which are later interpreted for various applications. The complex problem of extraction and interpretation of human motion presents several aspects each receiving specific attention by various researchers. In this thesis, we aim to contribute towards markerless multiple-view human motion capture or tracking (estimation of human motion information from video sequences-*extraction*) and classification (identifying the type of action from extracted information-*interpretation*). In this chapter, we present the applications of human motion analysis, motivations and scope of our research, outline of the contributions and finally, the structure of the thesis.

## 1.1 Applications of Human Motion Analysis

Human motion analysis has a wide array of applications in the areas of movies, computer games, biomedical applications, sports, security and surveillance. The most well-known application of motion analysis is in games and movies. Human motion capture, here, involves the digital recording and 3D representation of human motion. Special effects in many films like Avatar$^{TM}$, Lord of the Rings$^{TM}$ and computer animated films like Polar Express$^{TM}$ rely on human motion capture data to animate their characters (Figure 1.1). Typically the motion of actors are digitally captured in studios, using marker-based motion capture systems, and transferred to computed-generated figures for realistic animation. Similarly computer games also use the captured human motion information to create realistic character animations in the game [74].

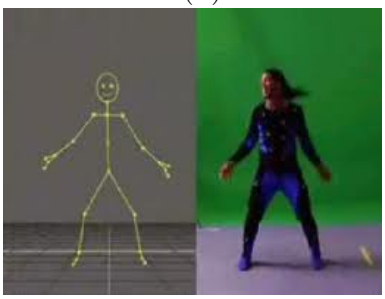
Gait analysis is an important example of biomedical application of human motion analysis. In gait analysis the degree of change in patient conditions with arthritis or strokes can be measured from the patient's motion information history [74]. Similarly, it can be used to monitor the rehabilitation progress of a patient after surgery or treatment. It is also used to evaluate different prothesis and identify incorrect postures [24].

Another application domain is sports coaching, where the motion of athletes can be captured and analyzed. Motion captured from leading athletes can be used as a basis for comparison, evaluation of other athletes. Moreover they can be used to detect and prevent mistakes, which could potentially degrade performances or cause injuries. Cricket, tennis and golf are some of the sports which can benefit from such coaching tools [68]. Another sport-based application is in post-match game analysis using the extracted athlete information from the video broadcast of a game.

In surveillance and security, analysing human motion is important to detect sus-


(a) Polar express$^{TM}$

(b) Pirates of the Caribbean$^{TM}$



(c) Fifa 2007$^{TM}$

Figure 1.1: Example of human motion capture, being applied in (a) animated features, (b) movies and (c) computer games.

picious behaviour or atypical motions, and trigger some alerts[29, 74]. Moreover, the gait of a person can be used as a biometric signature for security systems [74]. In human-computer interface-based applications for smart-homes, where the recognition of human gesture and motion can be used as communication tool for interaction with homes or offices, for example, switching on the lights, heaters or air-conditioners allowing the user to interact easily with their environment [18, 74].

## 1.2 Motivation and Scope of our Research

Vicon systems [139] are the established industrial state-of-the-art technique in marker-based human motion capture for the extraction of human motion inform-

ation at high accuracy, making such systems ideal for animation, games and the film industry. However these motion capture systems are invasive, requiring the users to wear special clothing and markers, resulting in unnatural human motion [89]. Moreover, they require a complicated and time-consuming preparation. Finally they are very expensive, which makes them unsuitable for many practical applications. An alternate approach addressing the above issues is markerless motion capture.

A key component in markerless human motion tracking systems are cameras, which are typically low-cost and flexible devices. In recent years, technological advancements have resulted in cameras providing high-resolution human motion information, easier integrability with computers and multiple camera synchronisation, making markerless capture systems suitable for many applications. Ideally a markerless tracking system should be able to extract human motion information from "any" video sequences, independent of the environment or the user. However this general problem remains largely unsolved and the existing state-of-the-art markerless tracking systems are constrained to specific environments. Similarly our markerless human motion tracking (Chapter 3 and 4) and classification system (Chapter 5) are constrained to studio environments with multiple cameras, and assume only one subject in every sequence. Furthermore, we do not tackle the problem of background subtraction, with human figure (foreground) being extracted from chroma-keyed studio background or provided by public datasets like HumanEva [114], as readily available foreground information.

In our markerless multiple-view human motion tracking, the main challenges include the complex nature of human motion, style and speed variations among different subjects performing the same action at different time instants, high-dimensional search space, noisy information from the cameras and self-occlusion from the limbs. We formulate the full-body articulated tracking from multiple-view sequences as a non-linear optimisation problem solved using a powerful

swarm-intelligence algorithm, particle swarm optimisation (PSO). Our PSO-based tracker eliminates the need for sequence-specific motion model, thus functioning as a black-box system. The same algorithm with unmodified parameter settings is able to track different motions with no prior information. However the fixed parameter settings result in increased computational complexity for certain actions. We address this issue by proposing a modified PSO-based tracker, the adaptive PSO-based tracker (A-PSO), wherein information from tracking is used to vary the parameters online. Although we report good tracking results using these systems, a motion model can be used to improve the tracking accuracy and the overall robustness of the system, at the cost of limiting the number of actions tracked. We exploit prior motion information in the form of a low-dimensional subspace, a reduced dimensionality representation, which functions as a good approximation of high-dimensional data and forms the basis for our second tracking framework.

In our subspace multiple-view markerless tracking system, we learn the motion models of common actions in a low-dimensional subspace using charting, a non-linear subspace learning technique. Tracking takes place in the subspace using a modified particle swarm optimisation algorithm biased for subspace optimisation. Additionally we aim to reduce the computational cost associated with the previous system's hypothesis evaluation, by using shape context histograms-based descriptors as our feature descriptors instead of silhouettes. Finally in order to evaluate hypotheses in subspace, we learn the mapping from the learnt low-dimensional subspace to the the shape context histogram-based descriptors using multi-variate relevance vector machines (MVRVM). We demonstrate good tracking accuracy, where our subspace tracking system performs better than our black-box systems, while achieving comparable accuracy with similar existing state-of-the-art tracking system.

The final component of our human motion analysis system is the multiple-view

classification algorithm. Similar to human motion tracking, human motion classification is also an important problem in human motion analysis with similar challenges, including high-dimensional search space, the complex and unpredictable nature of human motion and the noisy image or motion information. Additionally, an important challenge in classification is the similarity between certain actions, for example a fast walk would be similar to a slow jog, making it difficult to classify such actions. We propose a multi-layered classification framework for multiple-view sequence of extracted human motion information, which are either the output of our markerless tracking system or obtained from marker-based human motion capture systems. Similar to our subspace tracking system, the classification is performed in the low-dimensional subspace learnt using charting. The main motivation of adopting a multi-layered classification scheme is the successive pruning of candidate actions at each layer with less demanding classification search, until only fewer actions with subtle variations remain in the final layer. Here a demanding classification search is performed using multi-dimensional dynamic time warping, a feature vector alignment algorithm. Additionally, we present two variations of our classification framework, In our first approach, each action is mapped to a single subspace space, learnt from multiple actions. In the second approach, instead of mapping multiple actions to a single subspace, each action is mapped separately to its action-specific subspace. We report good classification accuracies, on the HumanEva dataset and CMU motion capture dataset, which are comparable with the existing state-of-the-art classification systems. We also compare our two proposed classification variations on the same dataset, and report our observations.

## 1.3 List of our Key Contributions

In this section, we summarise the key contributions of our three human motion analysis systems. We highlight the major contributions with **bold** texts, while

the minor contributions are highlighted in *italics*. Additionally we provide a short description of each contribution.

- Markerless human motion tracking using particle swarm optimisation without any motion prior (Chapter 3)
  - **Particle swarm optimisation used for articulated full-body tracking.**
    - *A novel, hierarchical version of particle swarm optimisation algorithm (H-PSO) is used for full-body markerless human motion tracking.*
    - *A guiding-cylinder scheme is proposed for the hypothesis evaluation in H-PSO, providing spatial and temporal constraints and reducing the computational complexity.*
    - *An adaptive version of H-PSO is proposed, wherein the system parameters are automatically varied online based on the accuracy of tracking, thus reducing the computational complexity.*
- HPSO and APSO function as a model-free system with fixed system parameters, automatically initialise in the first frame of the video sequence and importantly addresses the issue of divergence, whereby the system is able to recover after a wrongly estimated pose.
- Markerless human motion tracking with particle swarm optimisation using subspace learning-based motion prior (Chapter 4)
  - **Charting not previously used for subspace human motion tracking**.
  - **Particle swarm optimisation not previously used for subspace tracking**.
  - **A modified particle swarm optimisation, specific for subspace tracking and called subspace PSO is proposed.**
- Our proposed charting-based subspace tracking framework automatically initialises, recovers online from wrong estimates and avoids divergence, due to a nearest-neighbour retrieval scheme present in our tracking framework.
- Markerless human motion classification with multi-layered classification framework (Chapter 5)

  - **Charting not previously used for human motion classification.**
  - **Estimating the minimum length of skeletal features required for accurate human motion classification.**

    * Derivation of sequence of key-poses from the human action subspace.
    * Comparison of multiple subspace and single subspaces for human action classification.

- Our proposed multi-layered human motion classification framework reports good classification accuracy with efficient layered pruning of candidate action sets.

## 1.4 Outline of the Thesis

In this chapter, we have provided a brief overview of human motion analysis along with its application. Additionally, we introduce the work done in the thesis. In Chapter 2, we discuss the related approaches to human motion analysis. Chapter 3 deals with our hierarchical particle swarm optimisation-based markerless human motion tracking system. We provide a detailed comparison of our system with the particle filtering paradigm using our experimental results. Additionally, we also demonstrate our system behaviour to detailed system parameter changes. Finally, we explain our proposed adaptive version of H-PSO (A-PSO) and evaluate its tracking accuracy. In Chapter 4, we present our proposed subspace tracking algorithm, where we exploit the prior motion information, by learning the low-dimensional subspace using charting, and estimate the human pose using a modified particle swarm optimisation. In Chapter 5, we describe our multi-layered charting-based classification with experimental results on HumanEva and CMU mocap dataset [27], used for comparison with existing state-of-the-art classification algorithms. Finally we perform a detailed analysis of the system behaviour with system parameter changes. Finally in Chapter 6,

we present our conclusions and possible future extensions. An overall layout of the thesis is shown in Figure 1.2.

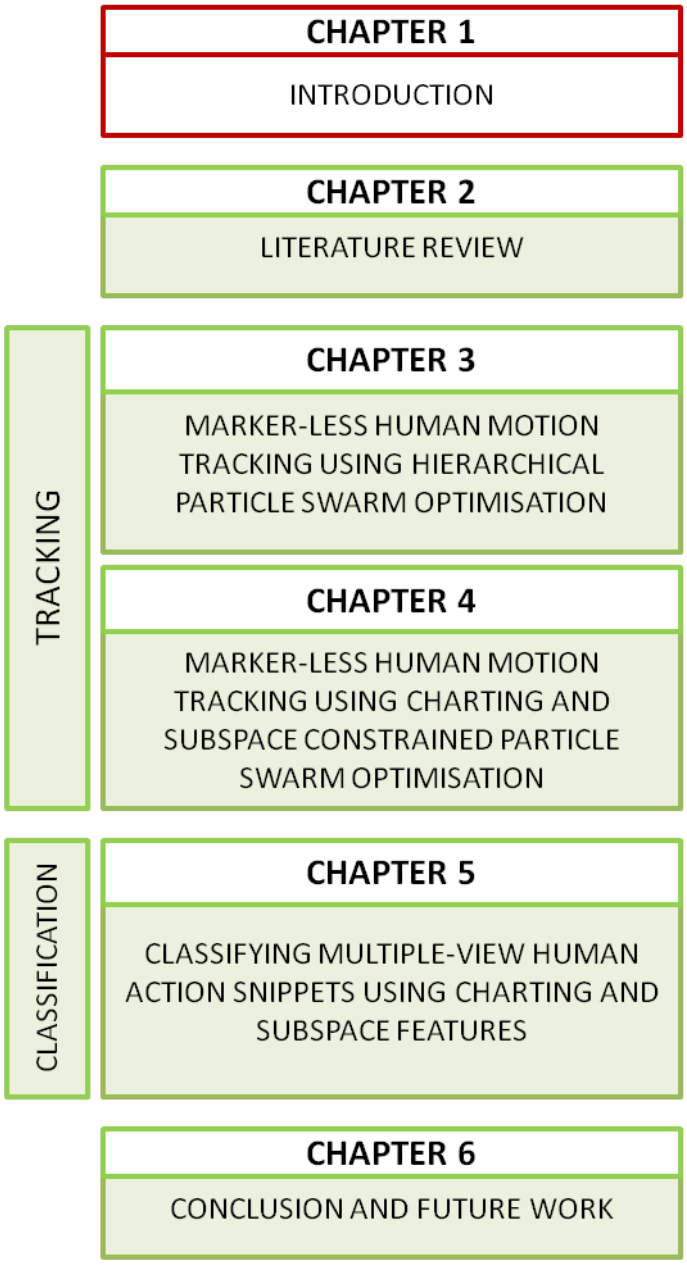



Figure 1.2: Thesis chapter layout

# Chapter 2

# Human Motion Analysis: Literature Review

## 2.1 Introduction

Articulated human motion tracking and classification is one of the most challenging problems in computer vision, and remains unsolved in its generality. The challenges can be attributed to the generally unpredictable and often complex nature of human movements, of self-occlusions created by limbs, of the high-dimensional search space induced by the skeletal models needed (between 20 and 40 degrees of freedom), shape variations existing among humans, and segmentation in non-studio applications. The literature of articulated human motion tracking and classification has grown very rapidly and in this chapter we present an overview of different motion analysis techniques. Firstly, we will present a brief overview of existing commercial motion capture systems. Next, we will provide a detailed survey of video-based markerless human motion tracking approaches and, finally, a detailed review of the human motion classification systems.

## 2.2 Commercial Motion Capture System

Typically commercial motion capture systems (mocap) use mechanical, electromagnetic or optical components to capture human motion information from subjects, without using video as an input. The systems essentially obtain the 3D position and orientation of the markers, which are attached to the subject. The 3D position and orientation of the different joints in the human body are then extracted from the captured marker positions. Some of them incorporate software to estimate the 3D position and orientation of the joints from those set of markers.

*Optical Motion Capture Systems.* Among the different types of mocap systems, optical motion capture systems are the most popular and widely used. Vicon system [139] and Qualisys [93] are examples of optical systems. These systems require the actor performing the motion to wear a special suit with markers, either reflective balls or pulsed light-emitting diodes, attached at various body positions. In reflective systems, infra-red light emitting diodes placed around the camera lens emit light, which are reflected by the markers and captured by the camera lens, fitted with infra-red pass filters [21, 149]. In case of pulsed systems, the infra-red light is emitted by the markers itself directly. Although the optical systems are accurate and widely-used, they are the most expensive technique .

*Mechanical Motion Capture Systems.* The Gypsy system of MetaMotion [69] and Physilog [86] are mechanical motion capture systems available commercially [21]. These systems have accelerometers and gyroscopes attached to the actors, which detect the body motion. The captured motion information are then either transmitted in real-time to the computer (Gypsy system) or recorded, processed

and transmitted at the end (Physilog) [131]. These systems are cheaper than other type of mocap systems, but they tend to be less accurate [21, 131, 149].

*Magnetic Motion Capture Systems.* The MotionStar from Ascension [7] and Liberty from Polhemus [63] belong to another category of motion capture systems measuring the magnetic field generated by markers or sensors, attached to the body. However this system is highly susceptible to noise, especially when multiple actors perform in the same environment causing interference of the magnetic fields [149]. Moreover, the markers in these systems move during capture, and often require recalibration, but magnetic mocap systems tend to be cheap [149].

The mocap systems discussed so far usually capture the 3D position and/or orientation of the markers, and require external software to infer the 3D joint angles of the articulated human body [131]. While Vicon provides the software along with their mocap system for this inference, other systems need separate softwares like MotionBuilder [76] to perform this inference task.

Although the commercial motion capture systems described so far are accurate and the established state-of-the-art, they have some serious disadvantages, in terms of cost, restrictiveness, and time-consuming experimental setups. All these factors compounded by the availability of cheap, high-quality cameras are responsible for the growth in interest for markerless vision-based human motion tracking and classification systems, which we review in the next section.

## 2.3 Video-based Markerless Human Tracking

Vision-based motion tracking systems, with cameras as sensors, continue to be an active research area attempting to address the issues with marker-based motion tracking. The problem of human motion tracking is significantly different and

(a) (b)

Figure 2.1: Examples of (a) occluded silhouette and (b) noisy silhouette with missing body part [8].

more difficult than the general object tracking problem, because of the human body structure and human motion dynamics. An important challenge in human motion tracking is foreground segmentation, difficulties include background noise and appearance variation among different human subjects. This has resulted in segmentation in itself being a separate, focused area of research. Since our work does not contribute to the human body segmentation literature, we do not provide a detailed report of the existing segmentation technique and refer the readers to [89, 74, 73]. Apart from typically noisy segmented human figures as shown in Figure 2.1, challenges include the generally unpredictable and potentially complex nature of human movements, self-occlusions created by limbs, occlusion from background objects, the high-dimensional search space induced by the skeletal models used (between 20 and 40 degrees of freedom), and appearance variations existing among humans and a large body of work in human body tracking exists focusing on addressing these issues. We next provide a brief overview of popular tracking algorithms, followed by a classification of different human motion tracking algorithms.

### 2.3.1 Human Motion Tracking Algorithms

Popular tracking algorithms used in markerless human motion tracking include filtering techniques like the classic Kalman filter and its variations [142, 70], mean shift [23], multiple-hypothesis tracking [61], importance sampling approaches like the particle filter (PF) [30, 92] and variations, iterative closest point (ICP) and variations [72], constrained non-rigid factorization [105], Markov models [85] and gradient boosting [12]. Wang and Rehg [142] reported a comprehensive comparison of particle filter algorithms for articulated figure tracking and proposed a new algorithm, the optimised unscented particle filter. Tweed and Calway [129] report a variation of particle filter with the introduction of bindings or subordination amongst particles, enabling the algorithm to handle multiple occlusions. Gradient-based methods have also been used to estimate pose and track articulated human figures in multiple-camera sequences. For example, Choo and Fleet [22] report a filter using hybrid Monte Carlo (HMC) and multiple Markov chains to generate samples from the target posterior distribution. The filter explores the state space rapidly, generating a substantially reduced number of particles compared to conventional PF.

In recent years, evolutionary optimisation approaches, like genetic algorithms, have been reported for articulated pose estimation from video sequences [147, 82]. In our work, we use particle swarm optimisation (PSO) an evolutionary optimisation approach introduced in [55], where a population of particles explore simultaneously the search space generated at each time instant. Each particle has a position and a search velocity associated with it. The search behaviour of the particles is governed by their interaction and designed originally to simulate the swarming behaviour of bird flocks in their search for food. PSO has been growing in popularity in a number of research areas as a technique to solve large, non-linear optimisation problems, as shown in the recent survey by Poli [87], but its applications to computer vision are still rather limited. To the best of

our knowledge, our work is the first application of PSO to full body articulated human motion tracking.

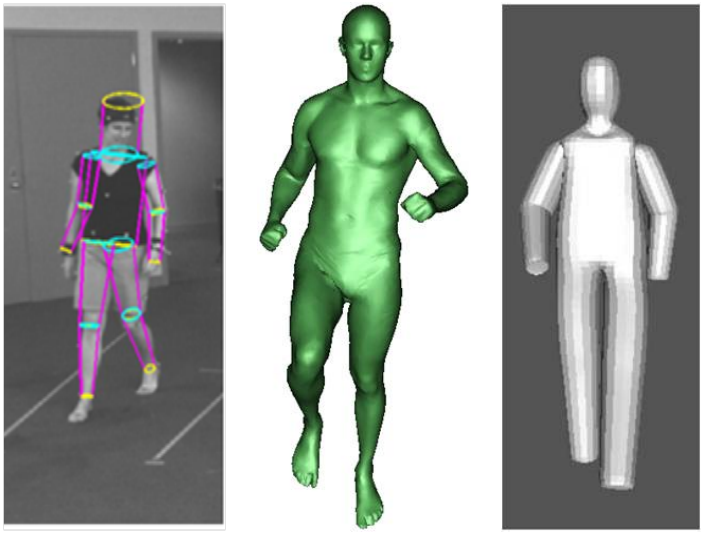
### 2.3.2 Classification of Human Motion Tracking Systems

In this section, we present two different classification methodologies for human motion tracking literature. In the first methodology, the human motion tracking literature is divided into two categories, generative and discriminative tracking human methods. In the second methodology, we categorise the existing literature based on the algorithm parameters.

#### 2.3.2.1 Generative and Discriminative Methods

Methods for markerless, articulated motion tracking are frequently classified as generative or discriminative.

*Generative Methods* use the analysis-by-synthesis approach, whereby a pose hypothesis is applied explicitly to the three-dimensional body model (skeleton and surface) to generate synthetic images (or features or parts there of) for each camera, and the real and generated images compared within an appropriate likelihood function to evaluate the quality of the pose hypothesis [30, 54, 85, 115, 72, 16, 112, 98]. The full-body model, used in generative methods, typically consists normally of an articulated skeleton capturing pose, and surfaces "fleshing out" each limb of the skeleton. Very often simple geometric primitives like cylinders or cones are used, but more complex surfaces have been used in some cases [28, 50] (Figure 2.2). Balan et al. [9] use a stochastic optimisation based on annealed particle filter, where SCAPE model is used to generate images for likelihood evaluation of the hypothesis. Bandouch et al. [10] propose



(a) (b) (c)

Figure 2.2: Examples of full body 3D body models (a) Cylindrical body model [8], (b) SCAPE body model [9] and (c) Catmull Clark's subdivision model [51]

a system using hierarchical annealed particle filter and RAMSIS body model to generate images for evaluation. A summary of some representative generative tracking methods are provided in Table 2.1.

*Discriminative Methods* infer the pose directly from the image by either learning the mapping between the pose space and a set of image features [12, 116] or using an exemplar approaches, where human pose is infered from input images by matching the input images to a set of stored exemplars of image-based features [34, 75, 128]. A few representative techniques for learning the mappings between the pose space and image space include relevance vector machines, support vector machines [3], mixture of Gaussians [2] or mixture of Gaussian processes [133]. While generative techniques are easier, flexible and more accurate, they tend to be computationally expensive. Compared to generative methods, discriminative methods tend to be faster once trained properly. However in some cases they are not as accurate as generative models [80]. We provide a summary of a few recent discriminative tracking systems in Table 2.2, at the end of the chapter.

*Combinations of Generative and Discriminative* approaches have also been reported [113, 118, 9], combining the advantages of both the techniques. Typically, the

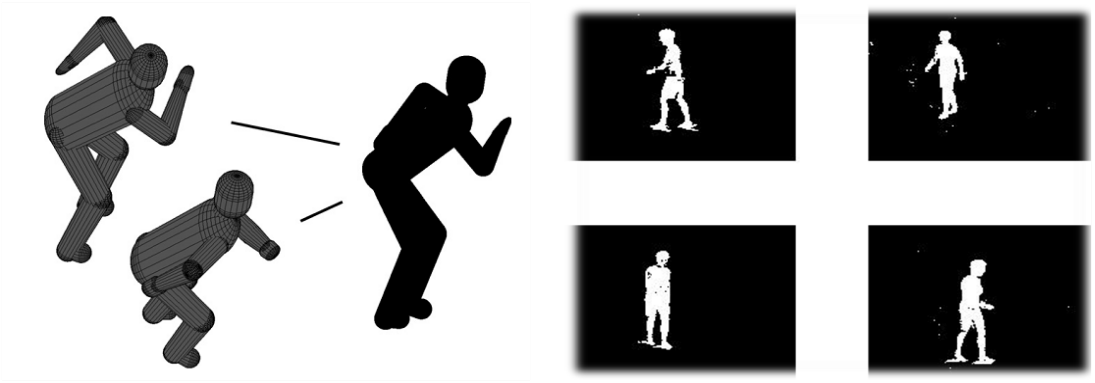
discriminative approaches are used to initialise the generative tracking framework or localise the search in every frame [40, 44].

#### 2.3.2.2 Literature Classification based on Algorithm Parameters

Articulated motion tracking solutions can also be classified using the parameters of the algorithm, including number of cameras, acquisition environment, and the use of priors in the form of motion models and/or search limits.

**Number of Camera Views** Based on the number of camera views, human motion tracking algorithms can be classified as either *single view* (*monocular*) or *multiple view*. A single camera tends to be cheap and can be setup in different environments. So monocular view algorithms can be used in uncontrolled environments, resulting in a wide range of applications [79, 42, 111, 94, 136, 84]. However, they provide less descriptive human motion information for tracking compared to multi-view techniques. Moreover, using a single camera results in self-occlusions and depth ambiguities (Figure 2.3 (a)). On the other hand, multiple camera systems provide rich human motion information, and simultaneously address self-occlusion and depth ambiguities [100, 19, 112, 9] (Figure 2.3 (b)). However, multiple camera systems tend to be expensive and are difficult to setup, as the cameras need to be synchronised. This limits the multi-camera systems to different applications, especially in outdoor environments.

**Acquisition Environment** Human motion tracking algorithms can be classified as *outdoor* or *studio environment*. Outdoor scenes are not controlled and can be noisy, due to the varying background and lighting conditions. Furthermore, in addition to self-occlusions, the persons are typically occluded by other objects



(a) (b)

Figure 2.3: (a) Depth ambiguity with monocular views [46] and (b) Depth ambiguity and occlusion can be avoided with multiple camera views [8]

in the scene [79, 94, 85, 19, 84, 136, 42]. This makes tracking human motion in outdoor environment challenging. On the other hand, in indoor studio environments, the scenes can be controlled with specific lighting conditions, ideal camera positions etc. However they are difficult to setup and expensive [30, 72, 9, 111].

**Constraints** Finally, human motion tracking algorithms can be classified based on the constraints used to estimate the pose from the video sequences. Apart from algorithms not using any constraint, which form a class on themselves, the literature of human motion tracking can be classified based on the type of search constraints and motion constraints.

*Search Constraints.* An important challenge in for the different search techniques in articulated human tracking is the high-dimensional search space. Many researchers have sought to reduce the complexity of high-dimensional search by either partitioning the search space [67] or by learning the joint limits [43]. The search space is partitioned, for example, according to the limb hierarchy [48, 67]. In hierarchical search, the poses of the body parts are estimated sequentially, each estimate constraining the possible configurations of subsequent limbs in the chain [70, 48]. An inherent problem with this approach is the need to estimate accurately the initial partition, as a wrong pose estimate for the initial segment

can distort the pose estimates for subsequent segments [10]. Considering this limitation of hierarchical schemes, adding a motion model to the tracking system would greatly constrain the search problem.

*Motion Constraints.* The key motivation behind using motion models is to constrain the very expensive or unfeasible search problem [19, 100, 85]. We can regard motion models for human motion tracking as instantaneous or extended. Instantaneous motion models consist of recursive equations predicting the next pose from previously estimated ones. The classic example is Kalman filtering (KF) [70, 65], extended subsequently by particle filtering and its variations [30]. Extended motion models, on the other hand, seek to describe whole actions (e.g., walking, sitting down) or behaviours [19, 100, 85, 136]. The rationale is that action models provide a context which strongly constrains pose expectation in the next frame. The price is a reduced generality, as this idea requires a pre-defined set of actions. In terms of extended motion models, a widely popular approach used in recent years is learning the low-dimensional subspace of the whole actions, using the learnt model, of typically <10D, to constrain the search, increasing the tracking accuracy at reduced computational cost. In Chapter 4, we propose such a tracking framework in the low-dimensional subspace learnt using charting, a dimensionality reduction algorithm.

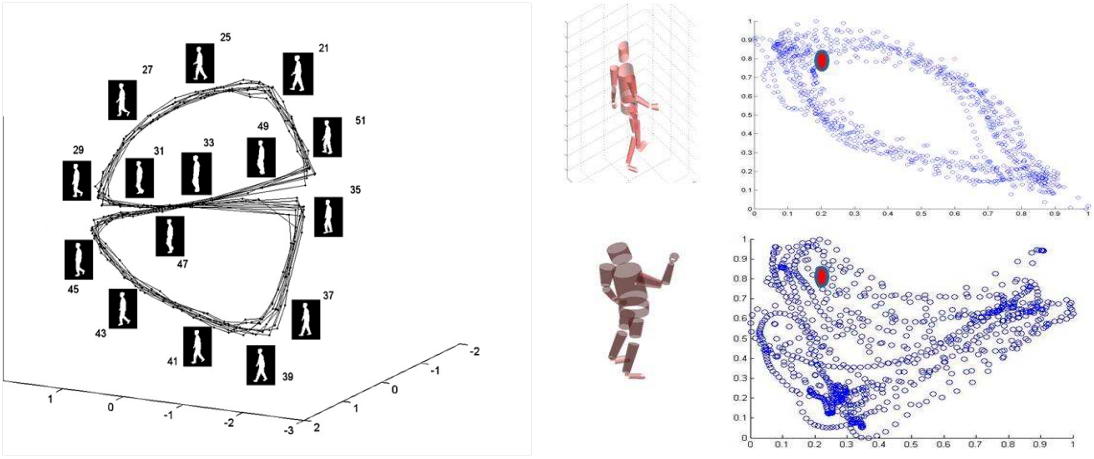
### 2.3.3 Placing Our Work in Human Motion Tracking Classification

In the context of our classification of human motion tracking, our first human motion analysis work (Chapter 3) presents a *generative* markerless multiple-view (**number of cameras**) human motion tracking algorithm using studio sequences (**acquisition environment**) using a hierarchical particle swarm optimisation (**hierarchical search constraint**) system functioning as a black-box system

(**no motion prior**).

## 2.4 Markerless Human Motion Tracking in a Low-Dimensional Subspace

Recent research in markerless human motion tracking has focused on approaches using learnt low-dimensional subspace of action models (Figure 2.4). Typically, low-dimensional subspace of human actions are learnt using a dimensionality reduction algorithm, and subspace tracking is performed. Tracking in the recovered subspace results in reduced computational complexity, increased accuracy and the possibility of real-time tracking, but at the cost of model switching [45]. The key component in subspace tracking approaches is *dimensionality reduction* or *subspace learning* algorithms. Several linear and non-linear dimensionality techniques have been proposed in the machine learning literature. Principal component analysis (PCA) is a linear dimensionality reduction technique, for example, used by Urtasun et al. [135] and Sidenbladh et al. [110]. However the mapping between the original pose space and subspace is in general non-linear, and linear PCA would fail to accurately learn the mapping. As a result, non-linear dimensionality reduction techniques like Isomap [124], locally linear embedding [101], Gaussian process latent variable model (GPLVM) [59], and local linear co-ordination (LLC) [102] are used to learn the human action manifold. Among the techniques described above, dimensionality reduction techniques like Isomap [124] or locally linear embedding [101] learn the non-linear mapping from high-dim space to low-dim subspace, but are not invertible, resulting in the need for separately learning the inverse mapping from the latent pose space to full-dim pose space using techniques like Bayesian mixture of experts (BME) [52], radial basis functions (RBF) [32] or relevance vector machines (RVM) [126]. On the other hand, techniques like Gaussian process latent variable model (GPLVM)



(a) (b)

Figure 2.4: (a) Learnt low-dimensional subspace representation of walk action[32] and (b) charting-based subspace (Chapter 4 and 5)

[26, 45, 137, 135, 96, 95], local linear co-ordination [61], and charting [15] formulate the inverse mapping directly within their dimensionality reduction framework. Similar to high-dimensional human motion tracking systems, the subspace systems can also be categorised into generative or discriminative approaches.

### 2.4.1 Generative Subspace Techniques

Similar to high-dimensional generative human tracking algorithms, subspace generative methods use the analysis-by-synthesis approach (Section 2.3.2.1). The central ideas in generative approaches are the following. Firstly, the pose in high-dimensional space are mapped to a low-dimensional subspace. Secondly, the low-dimensional subspace tracking technique is defined. Thirdly, a hypothesis evaluation scheme is defined. We next provide an overview of different generative subspace tracking systems, and discuss the popular techniques used in each module (three central ideas) described above.

*Dimensionality Reduction.* Firstly, based on an overview of recent generative subspace algorithms, GPLVM and its variations are the most widely dimensionality reduction technique. GPLVM is a non-linear dimensionality reduction technique, which learns a smooth mapping from the subspace to the full-dim pose space.

However the local distances in the pose space are not preserved by GPLVM [136], resulting in an inefficient tracking formulation. Back-constrained GPLVM solve this problem, by learning an additional mapping from full-dim pose space to subspace [45]. Another issue with the original GPLVM algorithm [59] is its limitation to smaller training sets, which is addressed in Urtasun et al. [138] by using sparse Gaussian processes. This variation is termed as the locally-linear GPLVM (LL-GPLVM).

*Subspace Tracking.* Secondly, tracking in subspace is largely performed by some flavour of particle filtering (PF) [45, 33, 39], multiple hypothesis tracker [61], or deterministic approaches [134].

*Hypothesis Evaluation.* Thirdly, the most popular approach for hypothesis evaluation involves mapping the subspace hypothesis to full-dim pose space creating the body models, and generating the synthetic image for evaluation with real images [96, 95, 58]. However this method of evaluation is computationally expensive. Alternatively, the hypothesis evaluation can be performed in the subspace itself without the need for any inverse mapping [40], specifically a mapping is learnt between the subspace of actions and the image space (e.g., silhouettes). Typically, image features like silhouettes are represented by low-dim descriptors, for example [40, 39], first reduce the silhouette to low-dim representation using vectorised descriptor-based on Gaussian mixture model. The bi-directional mapping between the vectorised descriptor and pose subspace is learnt using a Bayesian mixture of experts [40] and relevance vector machines, for evaluation. Jaeggli et al. [52] first learnt the pose subspace and appearance subspace using LLE and binary-PCA respectively. Next, the inverse mapping is learnt using RVM. Finally, for pose evaluation they learn the mapping from pose subspace to appearance subspace using RVM for evaluation.

In all the approaches discussed, the learnt low-dimensional subspace significantly increases the accuracy of the estimated pose. A further reduction in computa-

tional complexity and increase in accuracy can be obtained by the modelling of subspace dynamics. To this end, Gaussian process dynamical model (GPDM, [140]) was proposed as an extension of the GPLVM, where in addition to learning the low-dimensional subspace, variation in pose and appearance in subspace motion is also modelled.

### 2.4.2 Discriminative Subspace Tracking

In case of discriminative approaches, the pose is inferred from the image directly using two learnt mappings. First, from test image to learnt appearance representation (subspace); Second, from appearance subspace to pose space [32]. In Elgammal and Lee [32], LLE is used to learn the silhouette embedding and the mapping from low-dimensional silhouette subspace space to pose space is learnt using RBF. In Ukita et al. [130], the appearance subspace is learnt from a 3D visual hull generated from multiple-view silhouettes using GPDM. The pose subspace is learnt using PCA and, finally, the mapping from appearance to pose subspace is learnt using RVM. Similarly, in Sun et al. [121], a mixture of probabilistic PCA is used to learn the appearance subspace from silhouettes of image sequences. PCA is used to learn the pose subspace and RVM to learn the mapping between the appearance and pose subspace. Then, using the learnt models, the test silhouette is mapped to appearance subspace, then the embedded silhouette data is mapped to pose subspace using RVM, before estimating the pose using inverse PCA in the pose subspace.

Finally, in a few subspace tracking systems, the advantages of generative and discriminative have been combined together within the same framework [40, 4].

### 2.4.3 Placing Our Work in the Subspace Human Motion Tracking Classification

In the context of our classification of human motion tracking (Section 2.3.2), our second human motion analysis work (Chapter 4), presents a *generative* markerless multiple-view (**number of cameras**) human motion tracking algorithm using studio sequences (**acquisition environment**) and learnt action subspace (**extended motion model-constraint**)

In the context of the three central ideas in generative subspace human motion tracking (Section 2.4.1), firstly, we use charting, a dimensionality reduction algorithm, to learn the action model. Secondly, we use a modified version of the particle swarm optimisation for our subspace tracking. Thirdly, the hypothesis from PSO is evaluated without using any inverse mapping. We represent the multi-view silhouettes using low-dim shape context-based histograms representation and learn the mapping between the pose space and shape-context histogram space using multi-variate relevance vector machine. Given the learnt mappings, we map our hypothesis from pose space to shape-context histogram space for evaluation with test shape context histogram.

## 2.5 Video-based Human Motion Classification

Vision-based human action classification is the method of assigning an action label to an input video sequence, with applications in surveillance and security, gait analysis in sports, bio-medical and animation. The main challenge in human motion classification is modelling the complex, unpredictable human motion. Human action classification has been studied extensively in recent years. We give a brief overview here and refer the reader to [74, 90] for recent surveys.

The steps involved in video-based human motion classification include extraction

of discriminative features from video sequence, and assigning an action label to extracted features from a set of predefined action class labels. The extraction of discriminative features is essential for accurate human motion classification. Broadly, the human action classification literature can be classified based on dimensionality of feature. The derived features can be further classified as either *image-based features* [25] or *skeletal features* [66]. We next provide a brief summary about a few representative human motion classification algorithms in each literature class.

### 2.5.1 High-Dimensional Feature-based Classification

Of the two types of features, image-based ones are extracted directly from the video sequences for classification, while skeletal features are obtained from motion capture data or from tracking algorithms (Chapter 3 and Chapter 4). Image-based features are more widely used in human motion classification systems. The feature representations should, ideally, generalise over inter-person appearance variations for the same action, while simultaneously being discriminative enough to achieve accurate action classification.

*Image Features.* Bobick and Davis [14] use a silhouette-based feature known as binary motion energy image, obtained by summing the differences between successive frames in an action. The motion energy image provides a map of motion occurrence in the image. Wang et al. [143] obtain a silhouette-based representation, by applying a R-transform [41]. Blank et al. [37] represent the silhouette sequence information as 3D space-time features. In the work of Lin et al. [64], shape and motion information are extracted from the images and represented as action-prototype tree for efficient classification. Ning et al. [81] use appearance and position context descriptors as features; they train a discriminative conditional random field, termed latent pose estimators, where the observation layer of the random fields is replaced by an image-to-pose discrimin-

ative model. The above described approaches are 2D and depend on the camera view point. Weinland et al. [145] address this issue by creating a 3D voxel model obtained by combining silhouettes from multiple cameras. A recent development in human motion classification is the use of key-frames or prototypes to classify actions [90]. Sullivan and Carlsson [120] propose a shape-matching classification algorithm, where the test input sequence is compared with labelled key poses. Weinland and Boyer [144] propose a distance-based representation for silhouettes, using distance between action sequences and a set of discriminative key-pose exemplars.

*Skeletal Features.* While image-features are popular and have received great interest in the research community, there are a few human motion classification algorithms which are based on skeletal features. Zsolt et al. [47] propose such a system based on action primitives, which are sub-sequences of action derived from the complete action sequence. Lv et al. [66] model the dynamics of 3D joint position using HMM and learn a weak classifier for each joint position, which are then combined by the multi-class adaboost. A few algorithms combine tracking and classification [47]. Natarajan et al. [77] propose a similar system using conditional random fields.

### 2.5.2 Low-Dimensional Features-based Classification

The work discussed so far in human action classification use high-dimensional feature representations, resulting in increased computational complexity while assigning the action label. An alternative approach addressing computational complexity is learning *low-dimensional subspaces* for image or skeletal features. Techniques for identifying low-dimensional subspaces include local linear embedding [52], GPVLM [59] and its variations, locality preserving projections (LPP) [141], and local spatio-temporal discriminant embedding [53]. Similar to high-dim features-based classification, low-dim features-based classification can also

be further categorised into techniques using image features and skeletal features.

*Image Features.* Image-features-based classification algorithms are comparatively more popular than skeletal-features-based algorithms and widely used, with the pre-dominant feature being silhouettes. Wang [141] uses LPP with image silhouettes to learn the subspace for action classification. Chin et al. [122] learn silhouette subspaces using local linear embedding (LLE). Jia and Yeung [53] use local spatio-temporal discriminant embedding (LSTDE), where similar class silhouettes are mapped to nearby positions in the subspace. Niebles et al. [78] calculate patches of normalised space and time derivatives, and reduce the dimensionality using PCA after smoothing. Recently, Blackburn et al. [13] used Isomap to learn the subspace representation for smoothed silhouette sequences. Dynamic time warping is later used for matching test trajectories with database trajectories.

*Skeletal Features.* A few algorithms derive the low-dimensional subspace for skeletal features. In Han [57], the subspace for multiple actions are learnt from 3D skeletal features using hierarchical GPLVM (HGPLVM), a hierarchical extension of GPLVM. Additionally, there are subspace systems which again function as combined tracking-classification systems [96]. Chen et al. [20] use switched-GPDM to perform simultaneous action tracking and classification in the subspace. Similarly Jaeglli et al. [52], in addition to tracking in a subspace learnt using LLE, also perform action classification. Raskin et al. [96, 58] propose two systems for simultaneous action tracking and classification using HGPLVM [96] and GPLVM [58]. Recently, a few algorithms focus on enhancing the inter-class distance in the subspace. Such classification algorithms can be termed as discriminative classifiers [132, 109]. In Shyr et al. [109], a novel subspace estimation technique called *sequence kernel dimension reduction* is proposed for classification. Urtasun et al. [132] propose a discriminative GPLVM for action classification.

Based on our survey of the classification literature, we observe that image-based features are more widely used. This can be primarily attributed to the need to extract skeletal features either using a tracking algorithm or a motion capture system, whereas image-based features can be readily extracted from video sequences.

### 2.5.3 Placing Our Work in the Human Motion Classification Literature

In the context of existing human classification literature, our final motion analysis work presents markerless multiple-view human motion classification using *low-dimensional skeletal features.* Charting to learn separate action-specific subspaces (**low-dimensional**) from 3D joint angles (**skeletal features**). Moreover, we derive discriminative subspace motion patterns and key-frame-based representations from skeletal features, which are used in a multi-layered classification scheme.

Table 2.1: Summary of some representative generative human motion tracking approaches

| **Algorithms** | **Basic idea of the algorithms** |
|---|---|
| [9] | Tracking is performed using a stochastic optimisation based on annealed particle filter using Scape body model. |
| [10] | Hierarchical annealed particle filter, combining partitioned sampling and annealed particle filter is proposed for tracking using a RAMSIS body model. |
| [16] | Tracking is set up as simple linear systems based on differential motion estimation using the product of exponential maps and twist motions. Additionally a novel factorization technique is proposed to recover the kinematic chain model. |
| [17] | Tracking is performed using particle filtering constrained by an anthropomorphic walker, a stochastic controller that generates forces. |
| [19] | A particle filter-based algorithm using high-level behaviour, learnt using variable length markov models is proposed to track movements in real-time. Additionally a fast evaluation method based on volumetric reconstruction and blobs fitting is proposed. |
| [22] | A filtering based approach is proposed that uses hybrid Monte Carlo (HMC) to obtain samples in high dimensional spaces. An important component of the approach is the use of Multiple Markov chains that use posterior gradients to rapidly explore the state space, yielding fair samples from the posterior. |
| [35] | A multi-layer tracking framework that combines stochastic optimization (interactive simulated annealing), filtering, and local optimization is proposed. |
| [42] | An efficient Nonparametric Belief Propagation (NBP) algorithm is proposed in this paper for 2D articulated body tracking |
| [48] | A hierarchical partitioned particle filter (HPPF), is proposed using the prior knowledge of the structure of the human body. Additionally a motion model defined as action primitives, a sequence of consecutive poses, is used for stochastic prediction. |
| [54] | 2D human body part decomposition algorithm (HBPDA) that recovers all the 2D body parts of a subject by observing the shape of silhouette deformation, is proposed. The recovered 2D body parts are then integrated to obtain a 3D model of the subject. |
| [70] | Tracking is performed using extended Kalman filter, where the measurements are labeled voxel data. |
| [72] | Markerless motion capture system using a repository of visual hulls and articulated ICP. |
| [79] | A particle filtering based system, which estimates the pose from monocular image sequences. |
| [84] | Tracking framework is proposed using dynamic Bayesian network and switching linear dynamical system using 2D scaled prismatic model. |
| [85] | A tracking framework is proposed to handle real-world conditions including occlusions, error recovery, auto-initialisation. The action of different people are modelled using factored hierarchical hidden Markov model. |
| [100] | Tracking framework incorporating a motion prior modelled as twists. The tracked motion patterns are matched to training patterns, based on which prediction for the next frame is performed. |
| [110] | Tracking is formulated using a Bayesian framework integrated with stochastic optimisation. 3D pose is estimated from monocular images. |
| [142] | In this work, quantitative evaluation of different particle filtering human motion tracking algorithms is performed. Additionally a novel particle filter termed as optimal unscented particle filter is proposed. |

Table 2.2: Summary of some representative discriminative human motion tracking algorithms

| **Algorithm** | **Basic idea of tracking algorithms** |
|---|---|
| [12] | Pose estimation frameworks is proposed by learning the mapping from Haar features to pose space using multi-dimensional gradient boosting regression. |
| [105] | Tracking framework is formulated as a probabilistic inference problem. First, feature points are localised to landmark points in the human body such as elbows position, limb end-point position, before mapping to 3D pose using the probabilistic framework. |
| [116] | A mixture density propagation algorithm, based on conditional Bayesian mixture of experts, is proposed to estimate 3D human motion from silhouettes.of monocular video sequences. |
| [104] | Pose is estimated by learning the mapping from image space, comprised of a combination of histogram of shape context and histogram of local appearance context, using relevance vector machine. |
| [3] | Quantiative evaluation of different regression techniques, ridge regression, relevance vector machine (RVM) regression and support vector machine (SVM) regression, are evaluated for a single mapping from shape descriptors to pose space and RVM is found to perform the better than the other techniques. |
| [83] | Pose is inferred from histogram of gradient orientation using multiple support vector machine-based local linear regressors |
| [148] | SIFT-like descriptors are represented in a bag-of-words framework and the mapping function from image space to pose space is learnt using Gaussian process regression. |
| [133] | An online probabilistic regression scheme is proposed, where a local mixture of Gaussian processes are used to estimate the poses |
| [125] | Multivariate relevance vector machines are used to learn the mapping from image space to pose space and infer the pose from a given input image. |
| [38] | Bayesian mixture of experts is used to learn the mapping from novel silhouette descriptors, called Gaussian mixtures silhouette shape descriptor, to pose space to infer the pose. |
| [34] | An exemplar-based approach, where test motion exemplars are matched with training motion exemplars and pose is estimated from the match. |
| [106] | An exemplar-based approach, where matching is performed through brute force, using a variation of locality sensitive hashing for fast matching. |
| [128] | An exemplar-based approach using a probabilistic exemplar tracking model |
| [75] | An exemplar-based approach, where exemplar 2D views of the human body with different configurations and viewpoints, with labelled joint locations are stored. Given an input images, matching is performed with the stored views using shape context matching and pose is inferred. |

Table 2.3: Summary of some representative subspace human motion tracking approaches

| **Algorithm** | **Basic idea of the tracking algorithms** |
|---|---|
| [45] | Generative subspace technique, where first the latent pose space is obtained using back-constrained GPLVM. Next, the subspace is partitioned into clusters using unsupervised EM clustering and the temporal dependencies between the clusters are learnt using variable length Markov model. Finally, an efficient volumetric reconstruction algorithm, is used to evaluate the candidate pose obtained from the probabilistic subspace tracking algorithm. |
| [110] | Generative subspace technique, where PCA is used to reduce the dimensionality of a created action database, and particle filter is used as the tracking algorithm |
| [135] | Generative tracking using hill climbing approach in subspace, learnt by GPLVM. |
| [1] | Generative tracking framework, where the training data is clustered and PCA is used to reduce the dimensionality in each cluster. Next, the local dynamics are learnt and used within an pose inference algorithm defined in the subspace. |
| [52] | Generative tracking framework in subspace learnt using LLE, where the pose is inferred using a recursive Bayesian sampling algorithm. The evaluation of the hypothesis is performed using the pose subspace to appearance subspace mapping learnt using RVM. |
| [26] | Generative tracking framework in latent pose space learnt using hierarchical GPLVM and annealed particle filter is used to estimate the pose. |
| [137] | Generative tracking framework in subspace learnt using GPDM. The pose estimation is performed using an approximate recursive estimation technique. |
| [96] | Generative tracking framework, which is an extension of [58]. Tracking is performed in subspace learnt using hierarchical GPLVM. Additionally, the pose estimation is performed using a novel hierarchical annealed particle filter. |
| [61] | Generative tracking system using LLC to learn the subspace, where a simple multi-hypothesis tracker is used. |
| [33] | Generative tracking framework in a subspace, defined by torus, which is obtained as a combined representation of appearance and pose information. The pose is estimated using Bayesian tracking |
| [32] | Discriminative tracking framework, where LLE is used to learn the silhouette embedding and the mapping from low-dimensional silhouette subspace space to pose space is learnt using RBF |
| [121] | Discriminative tracking system, where mixture of probabilistic PCA is used to learn appearance subspace for silhouettes. RVM is used to learn the mappings between the appearance subspace and pose subspace, learnt using PCA. |
| [130] | Discriminative tracking framework similar to [121], where GPDM is used to learn the appearance subspace. Unlike similar discriminative subspace tracking systems, a particle filtering based scheme is employed in the appearance subspace to estimate the state, which is mapped to pose subspace and full-dim pose space using RVM and PCA respectively. |
| [40] | Combination of discriminative and generative methods, where GPLVM used to learn both the pose and appearance subspaces. Bayesian mixture of experts and RVM are used to learn mappings between the two subspaces. BME used to initialise generative tracker in pose subspace, discriminatively. The candidates from the pose subspace are evaluated through RVM mapping learnt from pose subspace to appearance subspace. |

Table 2.4: Summary of some representative human action classification algorithms

| Algorithm | Basic idea of algorithm |
|---|---|
| [25] | High-dim classification algorithm where actions are represented using 3D occupancy grids, derived from image features. |
| [66] | High-dim classification algorithm where actions are represented as set of feature spaces corresponding to joint motion. Additionally, the dynamics of each action class is learned using HMM. Finally several weak classifiers defined on HMM's observation probability are combined using multi-class Adaboost algorithm. |
| [64] | High-dim classification approach, where an action is represented as a sequence of prototypes. Specifically, an action prototype tree is learnt from joint shape and motion space. Using the prototype learnt, a lookup table of prototype-to-prototype distances is created. The actions are classified by sequence matching between the lookup table and action prototype tree. |
| [71] | High-dim classification scheme, where action is represented as a vocabulary forest of local motion-appearance features. The appearance features are obtained using various interest point detectors: MSER, Harris Laplace and Hessian-Laplace. Additionally, associated motion vectors are derived from optic flow. |
| [81] | High-dim classification algorithm based on conditional random field. A novel conditional random field is proposed, where the observation layer, defined on silhouette, is replaced with a latent pose estimator. |
| [62] | High-dim classification scheme is proposed, where human actions are represented as action graphs, where each node corresponds to a bag of 3D points, derived from sequences of depth maps. |
| [47] | High-dim combined tracking and classification scheme is proposed by introducing an action primitives model, which are sub-sequences of action derived from the complete action sequence. |
| [77] | High-dim combined tracking and classification framework using conditional random field, whose observation potentials are computed using shape similarity, while the transition potentials are computed using optical flow |
| [117] | High-dim recognised human motions based on discriminative conditional random field (CRF) and maximum entropy Markov models (MEMM), using image descriptors combining shape context and pairwise edge features extracted on the silhouette |
| [141] | Low-dim subspace classification framework, where dynamic shape subspaces of moving humans, represented by silhouettes, are learnt using locality preserving projections (LPP). Action classification is then achieved in a nearest-neighbor framework. |
| [57] | Low-dim subspace classification framework is proposed, where firstly, action subspace of skeletal poses are learnt using hierarchical Gaussian process latent variable model (HGPLVM). Next motion patterns are extracted from the subspace and used in a cascade CRF. Finally a trained SVM classifier is used to predict the action class. |
| [96] | Low-dim subspace combined tracking and classification framework using hgplvm and hierarchical annealed particle filter. |
| [20] | Low-dim subspace combined tracking and classification framework, where switched GPDM is used to learn a shared subspace from skeletal pose and silhouette moment features. |
| [109] | A novel dimensionality reduction called sequence kernel dimension reduction approach (S-KDR) is proposed to find a low-dim representation to perform efficient classification. |
| [122] | Low-dim subspace classification scheme, where the silhouette images are represented as a subspace learnt using LLE. Additionally, the learnt activity subspaces are extrapolated to a new test silhouette. |
| [53] | Low-dim subspace classification system is proposed for classifying human actions on embedded low-dimensional subspaces, learnt using a novel subspace embedding method, called local spatio-temporal discriminant embedding (LSTDE). |

# Chapter 3

# Markerless Human Motion Tracking using Hierarchical Particle Swarm Optimisation

## 3.1 Introduction

Tracking articulated human motion from video sequences forms the backbone of video-based human motion analysis with applications in virtual character animation, medical gait analysis, biometrics, human–computer interaction and others. In this chapter, we present our first human motion analysis algorithm, a generative full-body markerless human motion tracking framework from multi-view sequences. We formulate tracking as a non-linear optimisation problem which we solve using particle swarm optimization (henceforth PSO), a swarm-intelligence

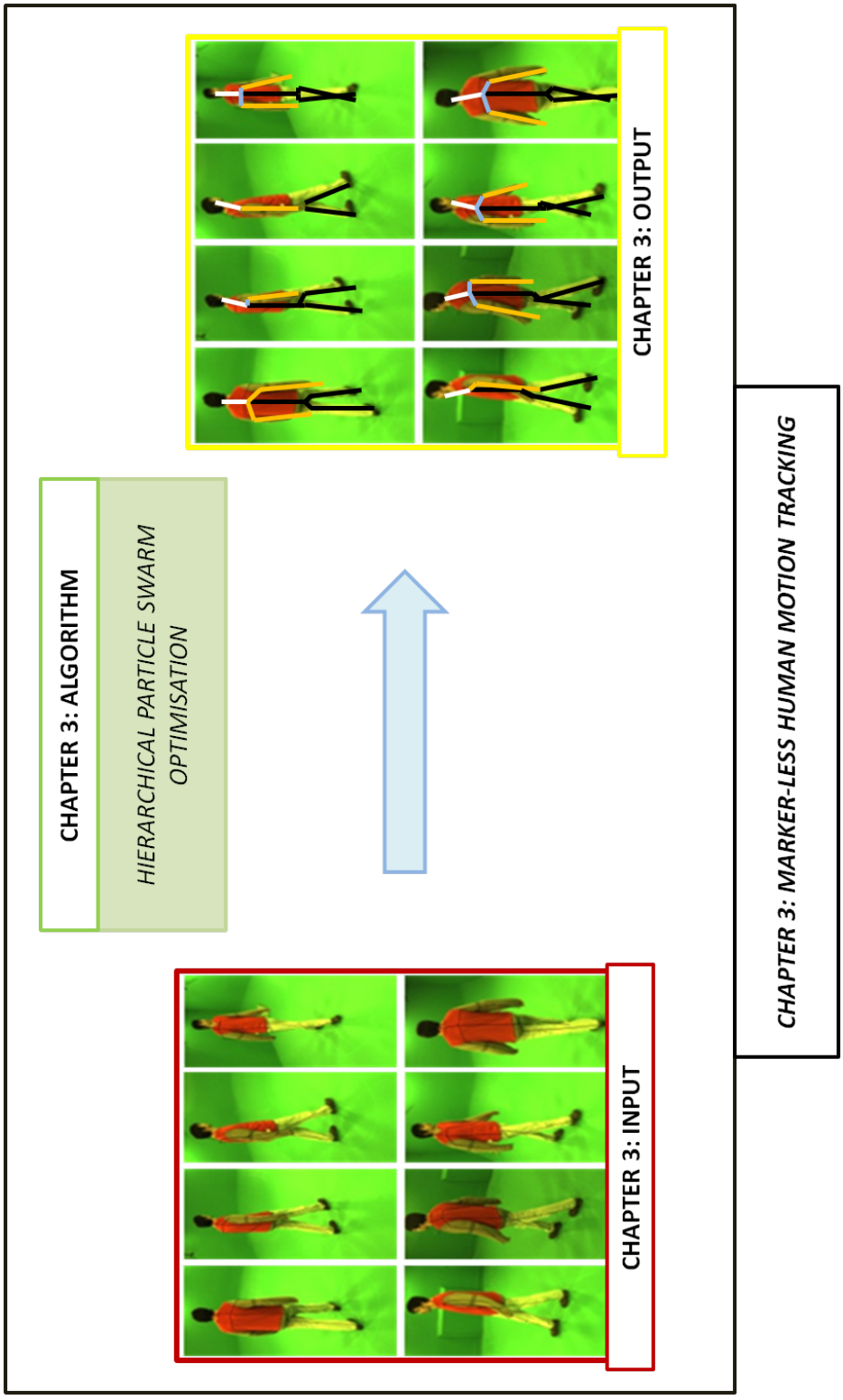



Figure 3.1: Overview of Chapter

algorithm with growing popularity [55, 87]. We show experimentally that a small-scale particle swarm, used with a standard body model and cost function, can produce tracking results which compare well or surpass those of recent, sophisticated algorithms based on particle filtering [30].

*Literature Classification Context.* In the context of classification of human motion tracking literature, defined in Section 2.3.2, we present a *generative* markerless multiple-view (**number of cameras**) human motion tracking algorithm using studio sequences (**acquisition environment**) using hierarchical particle swarm optimisation (**hierarchical search constraint**) based system functioning as a black-box system (**no motion prior**).

*System Overview.* In this chapter, we present our first human motion analysis algorithm, a markerless full-body articulated human motion tracking algorithm formulated for multi-view video sequences acquired in a studio environment, using a novel, hierarchical version of the PSO algorithm, called H-PSO (for hierarchical PSO), overcoming the limits of the popular particle filtering (Section 2.3.1) (henceforth PF) applied to articulated body tracking. Firstly, it removes the need for a sequence-specific motion model: the same algorithm with unmodified parameter settings is able to track different motions with no prior knowledge of the motion itself, producing results comparable with or superior to PF and related approaches. Secondly, HPSO addresses the problem of divergence, whereby the system is able to recover after a wrongly estimated pose. Divergence is sometimes combated by introducing additional, higher-level motion models [45] devising accurate predictions in the presence of known types of motions. In contrast, our tracking approach is designed to recover efficiently from an incorrect pose estimate and continue tracking without motion models. Thirdly, using the same mechanism deployed to recover from an incorrect pose estimate, HPSO estimates the first-frame pose with remarkable robustness, starting the search from a canonical pose.

HPSO extends our previous work on upper-body static pose estimation (no tracking) from multiple still images in videoconferencing-like scenes [50], by propagating the information from the previous instant (location of optimum at convergence) to the next instant and initializing the search around it (tracking). In order to ensure a fair quantitative comparison of HPSO and PF-based approaches, we use the computational framework provided by Brown University [8] to evaluate articulated full-body tracking algorithms using multi-view sequences. This package includes an implementation of PF and APF. We implemented our tracking approach within their framework by substituting the PF code with our HPSO algorithm and our implementation of partitioned sampling annealed particle filter (PSAPF). All other parts of the original implementation were kept the same.

We report a comprehensive and comparative experimental evaluation of HPSO. First, we report results of experimental comparisons of our algorithm with the particle filter (PF), the annealed particle filter (APF) and the partitioned sampling annealed particle filter (PSAPF) using the computational framework provided by Balan et al. [8]. HPSO accuracy and consistency are better than PF and compare favourably with those of APF and PSAPF, outperforming it in sequences with sudden and fast motion. Second, we analyse the effect of different cost functions. Third, we test the behaviour of the algorithm against variations of parameters and settings, specifically number of particles, number of cameras, model hierarchy, and localization of search for limb-specific pose estimation (guiding cylinders). While the hierarchical PSO (HPSO) approach, successfully estimated a wide range of different motion with a fixed set of parameters resulting in an unnecessary overhead in computational complexity. We address this in an adaptive approach, called APSO, which preserves the black-box property of the HPSO in that it requires no parameter value input from the user. Instead, it adaptively changes the value of the search parameters online, depending on the quality of the pose estimate in the preceding frame of the sequence. We compare the adaptive hierarchical PSO with HPSO and report the results. Finally, we

compare the adaptive hierarchical PSO with HPSO and report the results.

*Chapter Layout.* This chapter is organised as follows. Since we compare our proposed algorithm with particle filtering, we give a brief introduction to particle filtering, annealed particle filtering and partitioned sampling annealed particle filtering in Section 3.2. Section 3.3 presents the general PSO algorithm. Section 3.4 presents the genetic algorithm and simulated annealing. In Section 3.5 we present a discussion of PSO behaviour. Section 3.6 describes the body model and cost function used in our tracking approach while Section 3.7 presents the HPSO algorithm. Section 3.8 addresses the issue of adaptively setting the HPSO parameters to reduce the computational complexity. Section 3.9 reports the results of our experimental evaluation. Finally, Section 3.10 offers some conclusions and ideas for future work.

## 3.2 Particle Filtering

### 3.2.1 Standard Particle Filter

We briefly revise particle filtering (PF) as the basis of several recent articulated body trackers, and the main solution we use for comparative experiments. In PF, the tracking problem is formulated in a Bayesian framework: the goal is to estimate the posterior probability density function (pdf) $p(\mathbf{X}_t|\mathbf{Y}_{t:1})$ of the state $\mathbf{X}_t$ at time $t$ given a sequence of observations $\mathbf{Y}_{1:t}$ until that time instant.

The pdf is obtained recursively using the state dynamics $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ and the image observation likelihood $p(\mathbf{y}_t|\mathbf{X}_t)$, which is used as the weighting function, $w(\mathbf{X})$. Using these distributions, the pdf is formulated as

$$p(\mathbf{X}_t|\mathbf{y}_t) = \int p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathbf{y}_t|\mathbf{X}_t)d\mathbf{X}_{t-1} \tag{3.1}$$

This pdf is approximated by a set of $N$ samples (called *particles*), each representing a particular instance of the state vector. Each particle is associated with a weight reflecting the estimate of the pdf value for the state that the particle represents. Weights are denoted with $(\mathbf{x}_t^i, \pi_t^i)_{i=1}^N$, where $\mathbf{x}_t^i$ represents the $i^{th}$ particle at time step $t$ and $\pi_t^i$ is the normalised weight of the particle (related to the estimated pdf value). At each time step, the particle set is propagated using the state dynamics. The propagated particle set is then weighted by the likelihood $w(\mathbf{X})$ and normalised. A new unweighted particle set is obtained by resampling; in this step, particles are drawn from the particle set according to their weights. The process runs once for every time step. A detailed introduction and pseudocode of particle filters can be found in [6].

PF moves beyond traditional KF as it deals with non-linear non-Gaussian (hence multi-modal) pdfs. A number of variations of the PF have been proposed for articulated body tracking, including the annealed particle filter [30] and the partitioned sampling approach [67]. A brief introduction to the latter two is given below.

### 3.2.2 Annealed Particle Filtermeters

Deutscher and Reid in subspace [30] introduced the annealed particle filter (APF) for articulated human tracking, in which simulated annealing is used to guide the particles towards the global optimum and reduce the risk of getting stuck in local optima. Simulated annealing is integrated into the particle filter framework by introducing a parameter $\beta_m$ (Eq. 3.2), which smoothes the original weighting function $w_m$ (Eq. 3.2) within a multi-layered search. Each layer corresponds to a different particle filter

$$w_m(\mathbf{X}) = w_m(\mathbf{X})^{\beta_m}. \tag{3.2}$$

Following the simulated annealing paradigm, the weighting function is smoothed in the initial layers, then becomes increasingly detailed. This is achieved by a set of values $\beta_m < ... < \beta_1 < \beta_0$, with $m$ the number of layers, similar to an annealing schedule. A diffusion covariance is used to scatter the particles at each annealing layer; the amount of diffusion decreases with each layer. A detailed description of the APF is found in Deutscher and Reid [30].

### 3.2.3 Partitioned Sampling

A well-known approach for reducing the complexity of search in many dimensions is the hierarchical decomposition of the search space into sub-spaces whenever these can be identified meaningfully within a given problem. Partitioned sampling, as proposed by [67] for hand tracking, hierarchically decomposes the search space into partitions, which are estimated independently of one another. Partitioned sampling obtains superior results over particle filtering, by applying the dynamics and an appropriate weighted resampling sequentially in each partition. The weighted resampling is used to obtain a new particle set, re-weighted with respect to an importance function, which is peaked in the same region, as the posterior restricted to the current partition. Additionally, the weighted resampling operation ensures the pdf is not altered. The algorithm can only be used when specific conditions hold [67]. In partitioned sampling the decomposed dynamics and weighted resampling operations are applied sequentially to each partition. The weighted resampling operation ensures more particles populate the peak regions of the posterior restricted to the partition. The joint observation likelihood in the final partition evaluates the complete search space and constructs the posterior pdf. Bandouch et al. [10] incorporate an APF within the partitioned sampling framework (PSAPF), which implies that an annealing-like iterative approach is adopted in the decomposed dynamics and importance function of each partition. PSAPF is used for estimating articulated human pose:

the pose of the torso is estimated before focusing the search on the limbs and head. This is formulated as a set of hierarchically coupled local annealed particle filters. This approach does result in better accuracy than APF.

Our system seeks to address the following three drawbacks of the PF approaches: (i) divergence: the inability to recover from wrong pose estimates and resume tracking correctly; (ii) the need for manual initialization; (iii) the need for a sequence-specific motion model.

## 3.3 Particle Swarm Optimisation

PSO is a swarm intelligence technique introduced by Kennedy and Eberhart [55]. The idea originated from the simulation of a simplified social model, where the agents were thought of as birds and the original intent was to graphically simulate the unpredictable choreography of a bird flock in their search for food. The original PSO algorithm was later modified by several researchers to improve its search capabilities and convergence properties. In this thesis we use the PSO algorithm with inertia introduced by Shi and Eberhart [107]. PSO has been growing in popularity in a number of research areas as a technique to solve large, non-linear optimisation problems, as shown in the recent survey by Poli [87], but its applications to computer vision are still rather limited. To the best of our knowledge, ours is the first application of PSO to articulated human body tracking. Zhang et al. [146] report an application of a variant of PSO, called sequential PSO, to box tracking in video sequences. The authors suggest, in fairly descriptive terms, that the PSO part of their framework could be regarded as multi-layer importance sampling, although the exact relationship between importance sampling and PSO has not yet been completely analyzed; we offer some observations in Section 3.5.3. Anton-Canalis et al. [5] and Kobayashi et al. [56] are other examples of work in which PSO has been applied to non-articulated object tracking. Systems using PSO to estimate upper-body human

pose with static frames [50, 51], and preliminary attempts to PSO tracking using stereo data [99] are reported. The work reported in this chapter hinges on an experimental analysis of our particle swarm search, HPSO, compared to recent PF-based approaches, and others with qualitative comparisons. In addition, this chapter differs from our previous work in several ways, including using video sequences instead of single frames, multi-view silhouettes instead of stereo data, and full-body model instead of an upper-body one.

### 3.3.1 PSO with Inertia

Assume a $d$-dimensional search space $S \subseteq R^d$ defined by a pair of constraint vectors $\mathbf{a}, \mathbf{b} \in R^d$, a swarm consisting of $N$ particles, each particle representing a candidate solution to the search problem and a cost function $f : S \rightarrow R$ defined on the search space. The $i$-th particle is represented as an $d$-dimensional vector $\mathbf{x}^i = (x_1, x_2, ..., x_d)^T \in S$ subject to $\mathbf{a} \leq \mathbf{x}^i \leq \mathbf{b}$. The velocity of this particle is also an $d$-dimensional vector $\mathbf{v}^i = (v_1, v_2, ..., v_d)^T \in S$. The best position encountered by the $i$-th particle so far (*personal* best) is denoted by $\mathbf{p}^i = (p_1, p_2, ..., p_d)^T \in S$ and the value of the cost function at that position $pbest^i = f(\mathbf{p}^i)$. The index of the particle with the overall best position so far (*global* best) is denoted by $g$ and $gbest = f(\mathbf{p}^g)$. The PSO algorithm can then be stated as follows:

1. **Initialisation**:

   - Initialise a population of particles $\{\mathbf{x}^i\}, i = 1 \ldots N$, with positions randomly within $S$ and velocities randomly within $[-1, 1]$. For each particle evaluate the desired cost function $f$ and set $pbest^i = f(\mathbf{x}^i)$. Identify the best particle in the swarm and store its index as $g$ and its position as $\mathbf{p}^g$.

2. **Repeat** until the stopping criterion is fulfilled:

- Move the swarm by updating the position of every particle $\mathbf{x}^i$, $i = 1 \dots N$, according to the following two equations:

$$\begin{aligned} \mathbf{v}_{t+1}^i &= \omega \mathbf{v}_t^i + \varphi_1(\mathbf{p}_t^i - \mathbf{x}_t^i) + \varphi_2(\mathbf{p}_t^g - \mathbf{x}_t^i) \\ \mathbf{x}_{t+1}^i &= \mathbf{x}_t^i + \mathbf{v}_{t+1}^i \end{aligned} \tag{3.3}$$

  where subscript $t$ denotes the time step (iteration).

- Ensure that $\mathbf{a} \leq \mathbf{x}^i \leq \mathbf{b}$. Search constraints are easily enforced through particle velocities. If the particle violates the search space boundary in some dimension, its position in that dimension is set to the boundary value and the corresponding velocity entry reversed.

- For $i = 1 \dots N$ update $\mathbf{p}^i$, $pbest^i$, $\mathbf{p}^g$ and $gbest$.

The stopping criterion is usually either a maximum number of iterations or a threshold on *gbest* improvement. The parameters $\varphi_1 = c_1 rand_1()$ and $\varphi_2 = c_2 rand_2()$, where $c$ is a constant and $rand()$ is a random number drawn from $[0, 1]$, influence the *social* and *cognition* components of the swarm behaviour, respectively. In line with [1], we set $c_1 = c_2 = 2$, which gives the stochastic factor a mean of 1.0 and causes the particles to "overfly" the target about half of the time, while also giving equal importance to both social and cognition components.

Parameter $\omega$ is the inertia weight which we describe in more detail next.

### 3.3.2 The Inertia Weight

The inertia weight $\omega$ plays an important role in directing the exploratory behaviour of the particles: higher inertia values push the particles to explore more of the search space and emphasise their individual velocity, while lower inertia values force particles to focus on a smaller search area and move towards the best

solution found so far.

The inertia weight can remain constant throughout the search, or change with time. In our work, we use a time-varying inertia weight. We model the change over time with an exponential function which allows us to use a constant sampling step while gradually guiding the swarm from a global to a more local search:

$$\omega(c) = \frac{A}{e^c}, \qquad c \in [0, \ln(10A)], \tag{3.4}$$

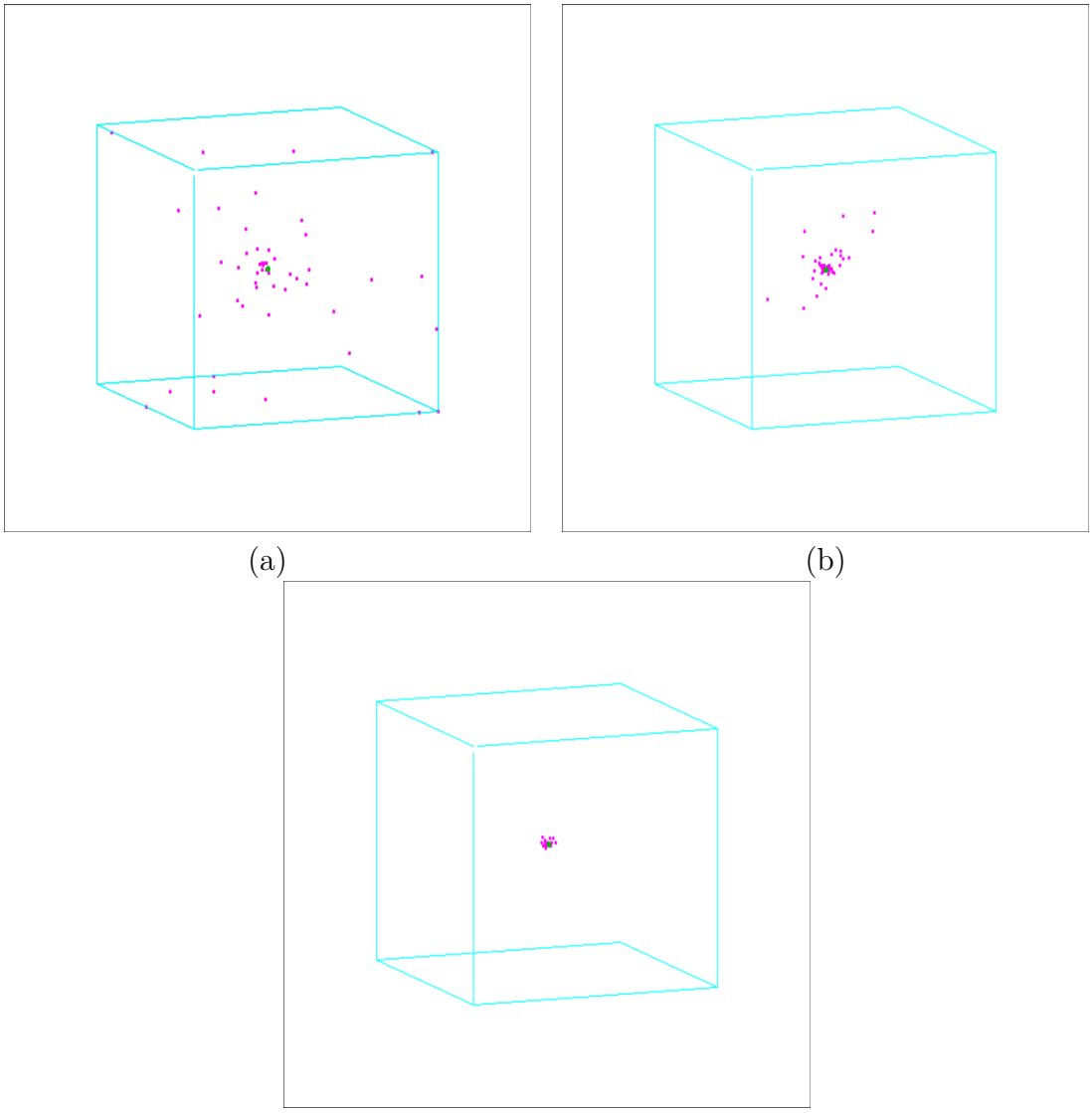
where $A$ denotes the starting value of $\omega$ when the sampling variable $c = 0$ and $c$ is incremented by $\Delta c = ln(10A)/C$, where $C$ is the desired number of inertia weight changes. The optimisation terminates when $\omega(c) < 0.1$.

As shown in Figure 3.2 (a), when the inertia is high, the particles explore larger portions of the search space (global search); with decreasing inertia, they settle around the globally best particle (local search).

## 3.4 Comparable Optimisation Algorithms

### 3.4.1 Genetic Algorithm

Genetic algorithm (GA) is an example of evolutionary algorithms generating solution to an optimization problem [82], with a technique inspired by natural evolution. Specifically, genetic operators such as mutation, selection, and crossover are used. In GA, a population of chromosomes (hypotheses) explore the search area and find the optimal solution at the end of a run, defined as the number of generations, or iterations. In the algorithm run, a population of chromosomes are randomly initialised, then evaluated using the defined fitness function. Next, based on the fitness of the chromosomes, the next generation is selected, followed by the application of crossover operation, creating a child by combining feature



(a) (b)

(c)

Figure 3.2: The effect of decreasing inertia, shown for a 3 DOF search space. The bounding box represents the search limits. The pink dot gives the *i-th* particle position, and the green dot, the global optimum for the frame considered. At high inertia values (a), particles explore large portions of the search space; particles overshooting the allowed boundary are placed onto the boundary for that iteration. The swarm localization effect for decreasing inertia values is shown in (b-c): fewer particles try to search outside the boundary, and search concentrates around the global optimum.

parameters from selected parents, and mutation operation, randomly changing a few feature parameters in each chromosome. This process is then repeated until the terminating condition is met.

### 3.4.2 Simulated Annealing

Simulated annealing (SA) is an optimisation technique [51], inspired by annealing process of metals. It seeks to avoid being trapped in local minima. Unlike gradient descent, SA accepts occasionally states which increase the fitness function, in addition to accepting states which decrease the fitness function. The annealing schedule uses a parameter termed as temperature $T$. Initially a global search is performed at high $T$, which is gradually decreased until a local search is performed. The annealing schedule is repeated until the terminating condition is met.

## 3.5 PSO Discussion

In this section, we present a discussion of PSO, and compare its behaviour with GA, SA and Bayesian filtering, before discussing convergence and its behaviour with multimodal fitness functions.

### 3.5.1 Comparison of PSO with GA

PSO is similar in a number of ways to GA. Both algorithms have a population of candidates; both update their population iteratively, while searching for the optimum stochastically. However, PSO differs from GA in the following ways. Firstly, PSO does not use genetic operators like crossover and mutation. Secondly, in GA, selection is used to choose chromosomes for each generation, while PSO does not have a selection parameter and the entire population is up-

dated in each iteration. Finally, in GA all chromosomes share information with each other, while in PSO only the global best particle's information is shared with the population. Compared with GA, in PSO all the particles tend to converge to the optimum solution quicker [31]. Additionally, in PSO, the search limits or constraints are directly integrated into the framework, whereas such a provision is not present in GA.

### 3.5.2 Comparison of PSO with SA

The annealing schedule of SA and varying inertia weight in PSO produce similar behaviour: an initial global search followed by decreased search area [51]. However, there exists a basic difference in the optimisation framework, primarily, PSO is a population-based optimisation scheme, while SA has only candidate hypothesis. Ivekovic et al. [51], demonstrate that PSO is able to estimate poses more accurately, than SA, which can be attributed to population of particles (candidates) in PSO.

### 3.5.3 PSO and Bayesian Filtering

It is a common misconception that the PSO algorithm is an implementation of a Bayes filter, in particular, the particle filter (PF), and that the PSO particles should therefore model a probability distribution over the available system states. The confusion usually arises from the choice of terminology: the particle filter uses *particles* to estimate the probability distribution over the system states, while the PSO uses *particles* to explore the cost function landscape. The PSO cost function does not have to be a probability distribution. The fitness associated with the PSO particle is therefore not the same as the PF particle weight. Additionally, each PSO particle also has its own velocity, a notion not present in the PF. Note that the PSO particle velocity is a property of the particle and not a component of the estimated state. A comparison of APF and PSO is shown in Figure 3.3.

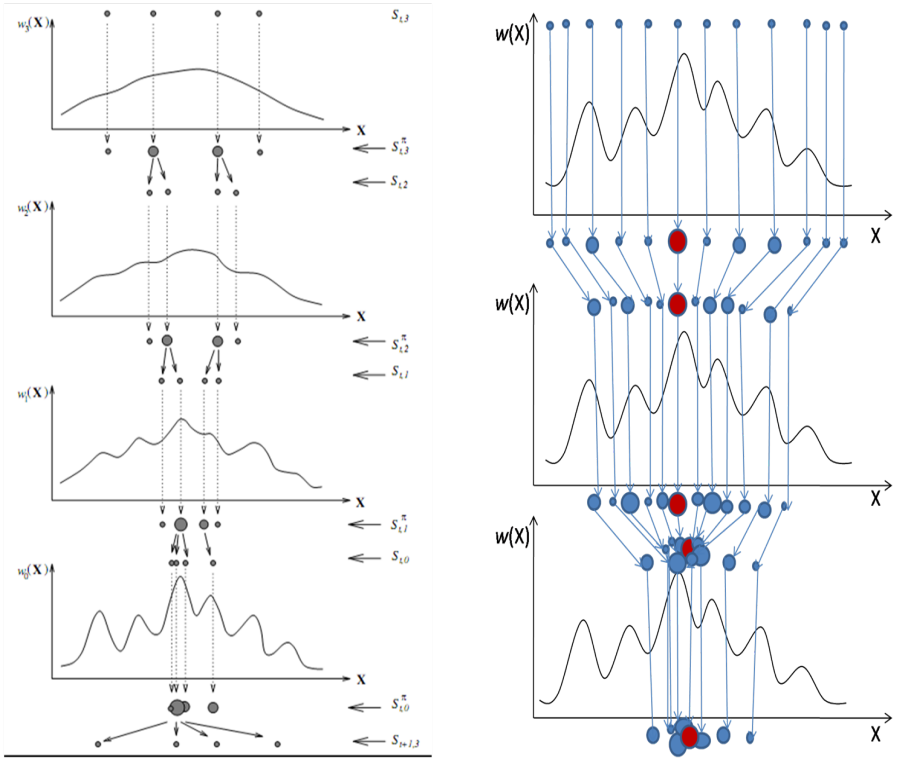



Figure 3.3: Illustration of pose estimation in a given frame for (left) APF and (right) PSO, where the red particle is the global best particle.

### 3.5.4 Convergence

Although the PSO algorithm appears deceptively simple, it is in fact a stochastic interacting particle system which is non-trivial to analyse. Its convergence depends, among others, on the choice of a cost function. The research on PSO convergence is still very much ongoing and the latest results by Poli [88] analyse the convergence behaviour of a stochastic PSO system under stagnation and give full account of the PSO sampling distribution, modelling PSO search behaviour. A number of experimental studies demonstrating the power of PSO search on specific problems have also been published recently [5, 56].

### 3.5.5 Multimodality

In our implementation, PSO particles always converge to a single state estimate

(the global optimum estimate). One reason is that the velocity update equation uses an inertia value parameter which is made to decrease over the iterations. As this happens, the attraction of every particle to the current global optimum increases until it eventually completely dominates the PSO behaviour, focusing the search of all particles and forcing them to converge onto a single estimate. Notice that the swarm could also be partitioned into sub-swarms, each using its own global best (i.e., over the sub-swarm). In this case, the algorithm would return a set of candidate optima at convergence. Our implementation does not support this option, as a single estimate seems to provide sufficient accuracy in our experiments.

### 3.5.6 Search Complexity

Unlike the Bayesian filtering scheme, PSO is an iterative search algorithm. Note that, although, PF is not an iterative algorithm, some variations like the APF [30] are iterative. Consequently, PSO, being an iterative search algorithm, does require more search effort to estimate the global optimum, compared to the PF algorithm. However, PF in order to estimate the global optimum requires a higher number of particles (order of hundred or thousand), while PSO requires fewer particles (order of ten). Furthermore, in our implementation, the increased PSO search effort does not result in an increase pose estimation time, owing to our proposed hierarchical evaluation scheme.

## 3.6 Body Model and Cost Function

This section summarizes the main features of the computational framework made available by Balan et al. [8], which we use in our experiments to enable a fair comparison with other tracking algorithms..

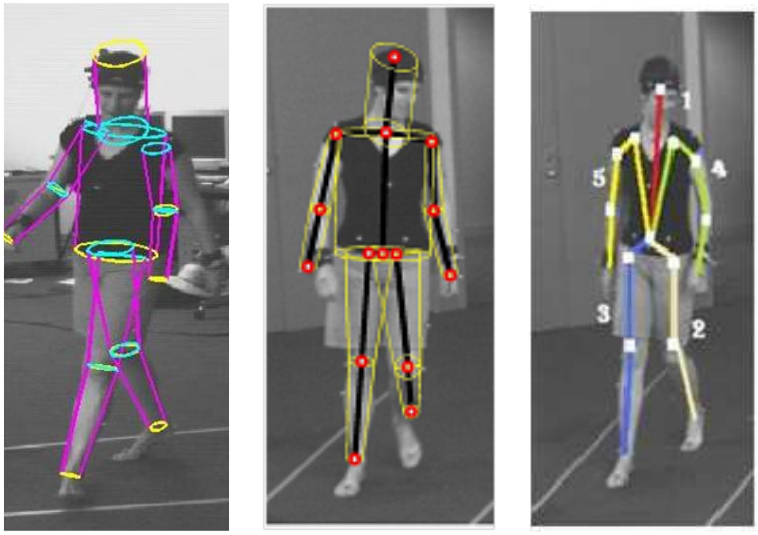
### 3.6.1 Body Model

The human body is modelled as a collection of truncated cones (Figure 3.4), and the underlying articulated structure is modelled with a kinematic tree containing 13 nodes. Each node corresponds to a specific body joint. For illustration, the indexed joints are shown overlaid on the test subject in Figure 3.4(b). Every node can have up to 3 rotational DOF, while the root node also has 3 translational DOF. In total, there are 31 parameters to describe pose and location of the full body (Table (3.1)). Each location of the particle in the swarm represents a 3D body model. The particles are evaluated by projecting the 3D body model onto the image as described in the next subsection.

The co-ordinates of a PSO particle in this 31-dimensional space represent a body pose and the position of the skeleton in the 3D world:

$$\mathbf{x}_i = (r_x, r_y, r_z, \alpha_x^1, \beta_y^1, \gamma_z^1, ..., \alpha_x^K, \beta_y^K, \gamma_z^K) \tag{3.5}$$

Here, $r_x, r_y, r_z$ denote the co-ordinates of the root of the kinematic tree, which identify the position of the entire body in the world coordinate system; $\alpha_x^k, \beta_y^k, \gamma_z^k$, $k = 1 \dots K$, are the rotational degrees of freedom of joint $k$ around the $x$, $y$, and $z$-axis, respectively. The equation does not strictly represent the state vector as many parameters have a fixed value (e.g., the elbow joint only uses 1 of the available 3 DOF). The actual state vector used in our experiments is given in Table (3.1). Considering the root position co-ordinates, $r_x, r_y, r_z$, the total number of DOF in the kinematic tree is $K + 3$, in our case 31, as said above.




(a) (b) (c)

Figure 3.4: (a) The truncated-cone body model. (b) Joint positions. (c) Kinematic tree

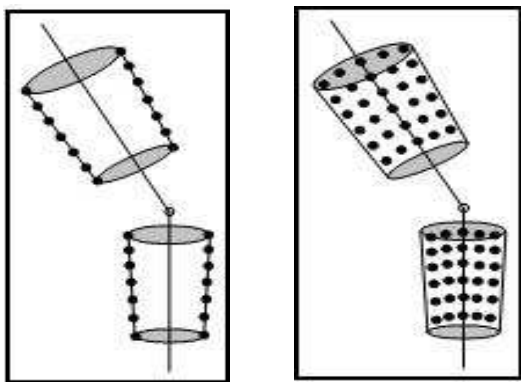
Table 3.1: Joints and their DOF

| JOINT (index) | # | DOF |
|---|---|---|
| Global body position (1) | 3 | $r_x, r_y, r_z$ |
| Global body orientation (1) | 3 | $\alpha_x^1, \beta_y^1, \gamma_z^1$ |
| Torso orientation (2) | 2 | $\beta_y^2, \gamma_z^2$ |
| Left clavicle orientation (3) | 2 | $\alpha_x^3, \beta_y^3$ |
| Left shoulder orientation (4) | 3 | $\alpha_x^4, \beta_y^4, \gamma_z^4$ |
| Left elbow orientation (5) | 1 | $\beta_y^5$ |
| Right clavicle orientation (6) | 2 | $\alpha_x^6, \beta_y^6$ |
| Right shoulder orientation (7) | 3 | $\alpha_x^7, \beta_y^7, \gamma_z^7$ |
| Right elbow orientation (8) | 1 | $\beta_y^8$ |
| Head orientation (9) | 3 | $\alpha_x^9, \beta_y^9, \gamma_z^9$ |
| Left hip orientation (10) | 3 | $\alpha_x^{10}, \beta_y^{10}, \gamma_z^{10}$ |
| Left knee orientation (11) | 1 | $\beta_y^{11}$ |
| Right hip orientation (12) | 3 | $\alpha_x^{12}, \beta_y^{12}, \gamma_z^{12}$ |
| Right knee orientation (13) | 1 | $\beta_y^{13}$ |
| TOTAL | 31 | |

### 3.6.2 Cost Function

The cost function for PSO measures how well a pose hypothesis matches the multi-view data from a set of synchronized cameras. The cost function proposed

by Balan et al. [8] is shown in Eq. (3.9); we shall refer to it as model weighting function. It consists of an edge-based part and a silhouette-based part.



(a) (b)

Figure 3.5: The sampling points obtained from the 3D cylinders for a) edge-pixel map along the contours of the cylinder and b) uniformly sampled inside the cylinders for the silhouette [30].

*Edge-based Part.* A binary edge map is obtained by thresholding the image gradients. This map is then convolved with a Gaussian kernel to create an edge distance map, which determines the proximity of a pixel to an edge. The model points along the edge of the truncated cones are projected onto the edge map and the sum of squared difference (SSD) between the projected points and the edges in the map is computed using

$$\Sigma^e(\mathbf{X},\mathbf{Z}) = \frac{1}{N}\sum_{i=1}^{n}(1 - p_i^e(\mathbf{X},\mathbf{Z}))^2 \tag{3.6}$$

where $\mathbf{X}$ are the projected model points, $\mathbf{Z}$ is the image from which the edge distance map is computed and $p_i^e(\mathbf{X},\mathbf{Z})$ represent the value of the pixel map at the projected model points.

*Silhouette-based Part.* A silhouette is obtained from the input images by statistical background subtraction with a Gaussian mixture model. A pixel map is then constructed, with foreground pixels set to 1 and background pixels set to 0. A predefined number of points on the surface of the 3D body model is then projected into the silhouette image and the SSD between the projected points

and the silhouette computed.

$$\Sigma^s(\mathbf{X},\mathbf{Z}) = \frac{1}{N}\sum_{i=1}^{n}(1 - p_i^s(\mathbf{X},\mathbf{Z}))^2 \tag{3.7}$$

where $p_i^s(\mathbf{X},\mathbf{Z})$ represent the value of the pixel map at $N$ projected model points, which are sampled from the surface of the body model. The configurations of the sampling points for the silhouette and edge-based part are shown in Figure 3.5. Examples of an edge-distance and silhouette map are shown in Figure 3.6.

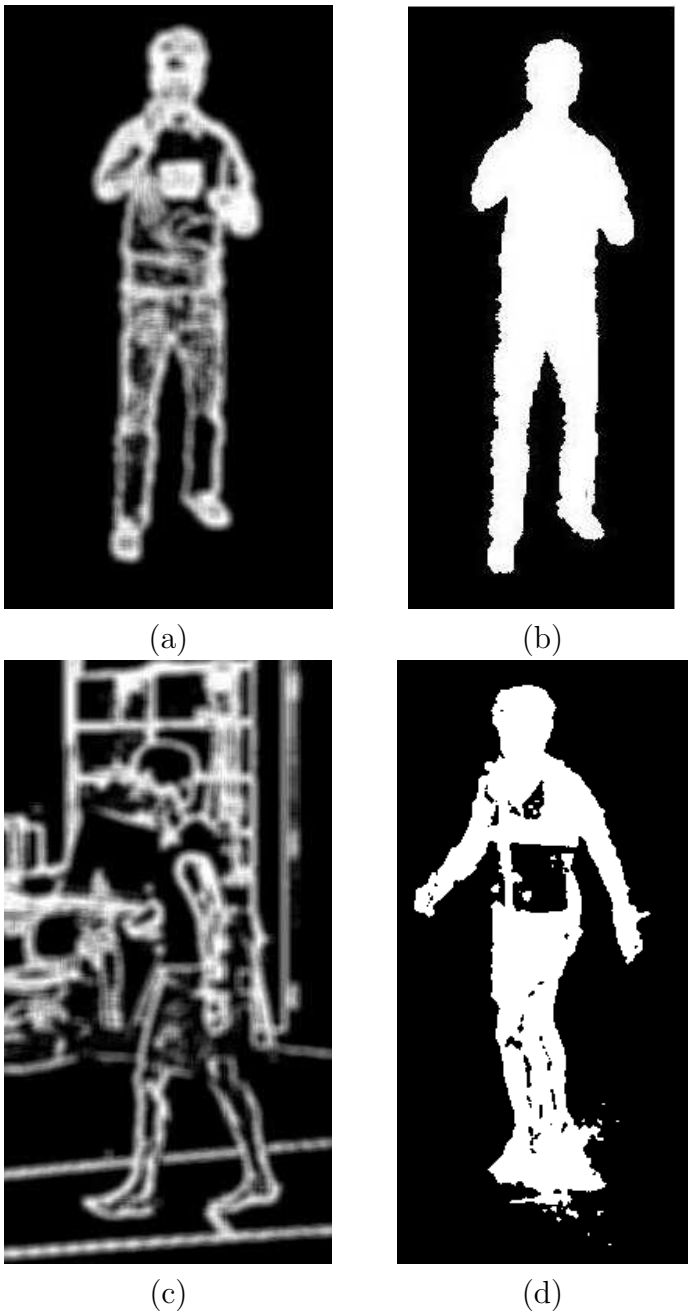
Finally, the edge and silhouette parts are combined to give the cost function value $f(\mathbf{x}^i)$ of the $i$-th particle :

$$f(\mathbf{x}^i) = \Sigma^e(\mathbf{X},\mathbf{Z}) + \Sigma^s(\mathbf{X},\mathbf{Z}) \tag{3.8}$$

and for multi-camera systems the cost function is obtained by summing over multiple (C) cameras,

$$f(\mathbf{X}^i) = \sum_{j=1}^{C}\Sigma^e(\mathbf{X}^i,\mathbf{Z}^j) + \Sigma^s(\mathbf{X}^i,\mathbf{Z}^j) \tag{3.9}$$

The accuracy of the pose estimate depends on that of the data computed from the observations, here edge and silhouette maps. Clean (a,b) and noisy (c,d) maps are shown for illustrations in Figure 3.6.



(a) (b)

(c) (d)

Figure 3.6: (a) A good edge-distance map: all edges found are within the target figure. (b) A good silhouette map: the contours follow closely those of the target figure, and the silhouette region is practically complete. (c) A noisy edge-distance map: some of the figure edges are missing (left contour of torso) and plenty of distracting edges are present. (d) A noisy silhouette map: the contour departs from that of the target figure (e.g., right foot area) and the silhouette region has significant holes. Figures (c) and (d) have been taken from [8].

## 3.7 HPSO Algorithm

The HPSO tracking algorithm consists of three main stages: initialisation, hierarchical pose estimation and next-frame propagation. We describe the three steps in detail next.

### 3.7.1 Initialisation

Initialisation is fully automatic. In the first frame of the sequence each particle in the swarm is assigned a random position within the constrained 31-dimensional search space $S$ and a random 31-dimensional velocity vector drawn from [-1.0,1.0]. In every next frame, the search is initialised by propagating the solution from the previous frame and sampling around it, as described later in this section.

### 3.7.2 Hierarchical Pose Estimation

Not unlike other algorithms, PSO becomes increasingly computationally intensive as the dimension of the search space increases [99]. To limit this effect, we search for the best pose hierarchically: the joints in the kinematic tree are optimised in a sequence, starting with the torso and proceeding towards the limbs. This follows the inherent hierarchical structure of the human body, where the configuration of the joints at higher levels of the kinematic tree constrains that of joints appearing at lower levels in the tree. As done commonly, we use this hierarchy to subdivide the search space into several sub-spaces, each containing only a subset of DOF. In our case, the hierarchy of the kinematic structure starts by estimating the position and orientation of the entire body, considered as a single, rigid object in the world reference frame. This result affects the configuration of every joint in the model. The kinematic tree then branches out into five chains: one for the neck and head, two for left and right arm, and two for left and right leg. The five branches of

the kinematic tree are shown overlaid on the test subject in Fig. 3.4(c). We split the search space into 12 different sub-spaces and correspondingly perform the hierarchical optimisation in 12 steps, detailed in Table 3.2. Furthermore, the estimate obtained for each sub-space is unchanged once generated. The sub-spaces are chosen so that only one limb segment at a time is optimised, and results are propagated down the kinematic tree.

*Guiding Cylinders.* At each step in the hierarchical search, the cylinders associated with the joints being optimised are the main optimisation targets (we call them primary cylinders (PC)) Additionally, adjoining cylinders which follow on the next hierarchical level are also projected to provide constraints to the search (guiding cylinders (GC)). For instance, if the pose of the upper arm is being determined by optimising the shoulder joint, the upper arm is projected as a primary cylinder and the lower-arm cylinder is projected as a guiding cylinder. Primary and guiding cylinders for each hierarchical step are shown in Fig. 3.18. Guiding cylinders provide an effective temporal and spatial constraint in obtaining the optimal pose for a limb. They provide an effective temporal constraint as the guiding cylinder for the current frame pose estimation is taken from the pose estimated in the previous frame (the only information propagated by HPSO). The spatial constraint is obtained from the kinematic tree structure, as the guiding cylinders are adjacent to the primary cylinder. The HPSO hierarchy (Table 3.2) defines which joint angles are estimated in a particular hierarchical step-the corresponding limb segment is modelled with the primary cylinder. The guiding cylinders, on the other hand, define which limb segments also have to be projected at that particular hierarchical step to facilitate the estimation of the angle values describing the primary cylinder configuration. The use of guiding cylinders does not change the cost function – it only designates which limb segments should be used to evaluate the cost function at a particular hierarchical level, in addition to the limb segments defined by the hierarchy given in Table 3.2, and so provides

Table 3.2: The 12 hierarchical steps of our HPSO full-body pose optimisation.

| | |
|---|---|
| **(Step 1)** Global body position: 3DOF: $r_x, r_y, r_z$ | **(Step 7)** Right lower arm orientation: 2DOF:$\gamma_z^7, \beta_y^8$ |
| **(Step 2)** Global body orientation: 3DOF: $\alpha_x^1, \beta_y^1, \gamma_z^1$ | **(Step 8)**Head orientation: 3DOF: $\alpha_x^9, \beta_y^9, \gamma_z^9$ |
| **(Step 3)** Torso orientation: 2DOF: $\beta_y^2, \gamma_z^2$ | **(Step 9)** Left upper leg orientation: 2DOF: $\alpha_x^{10}, \beta_y^{10}$ |
| **(Step 4)** Left upper arm orientation: 4DOF: $\alpha_x^3, \beta_y^3, \alpha_x^4, \beta_y^4$ | **(Step 10)**Left lower leg orientation: 2DOF: $\gamma_z^{10}, \beta_y^{11}$ |
| **(Step 5)** Left lower arm orientation: 2DOF:$\gamma_z^4, \beta_y^5$ | **(Step 11)** Right upper leg orientation: 2DOF: $\alpha_x^{12}, \beta_y^{12}$ |
| **(Step 6)** Right upper arm orientation: 4DOF: $\alpha_x^6, \beta_y^6, \alpha_x^7, \beta_y^7$ | **(Step 12)** Right lower leg orientation: 2DOF: $\gamma_z^{12}, \beta_y^{13}$ |

useful search constraints in case of occlusions (Section 3.9.1.1).

### 3.7.3 Next-Frame Propagation

HPSO propagates only a minimal amount of information between frames, and does not incorporate any motion model. Once the pose in a particular frame has been estimated, the swarm of particles is initialised in the next frame by sampling a Gaussian distribution centred in the current best estimate. The covariance of the Gaussian is set to a low value, in our case 0.01 for all joints, to promote temporal consistency. The lack of a prediction based on a dynamic model is motivated by two considerations: generality (we do not make assumptions on the type of motion) and the effectiveness of the swarm search, which can explore efficiently large portions of the search space starting from the initial distribution of particles.


Figure 3.7: The 12 steps in the hierarchical optimisation scheme are illustrated, where the yellow cylinders correspond to body parts being optimised (*primary cylinders*). Furthermore, the red cylinders in $(d, f, i, k)$ are the guiding cylinders, which constrain the search of the primary cylinders as explained in Section 3.9.1.1

### 3.7.4 HPSO Comparative Discussion

*Comparison with Hierarchical Filtering.* Although both HPSO and PSAPF are hierarchical algorithms estimating a state vector (pose), we observe two main differences. First, similar to PF, the set of particles in PSAPF aims to approximate a pdf, whereas HPSO does not. Second, PSAPF estimates within each sub-space are finalised after processing the final sub-space, whereas HPSO's estimate for each level of the hierarchy are unchanged once generated.

*Comparison with Combined Optimisation-Filtering.* Gall et al. [35] describe a multi-layer generative system combining global optimization, filtering and local optimisation. A 3rd-order autoregressive motion model is trained online and used to guide a stochastic optimisation. The first layer runs a global annealing search in the space of possible skeletal poses. The results are smoothed to reduce jitter, then used to refine the silhouette segmentation with a level-set algorithm. The improved segmentation supports a refinement of the pose estimation, achieved with a local search around the pose estimated in the first layer. Like HPSO, this approach can initialise independently with no external input. Unlike PSO, it predicts pose in the next frame with a 3rd-order autoregressive model, while HPSO carries over to the next frame only the position of the current optimal estimate of the state. In addition, Gall et al. require two segmentation steps at each frame, while HPSO uses a single step. The level of sophistication of Gall et al. is considerably higher than that of HPSO, yet results seem comparable by accuracy and other parameters (Section 3.9), suggesting that even a small-scale particle swarm search is capable of exploring a complex space with excellent results. For this reason HPSO does not employ motion models, either instantaneous (predictive equations) or global (action models).

## 3.8 Adaptive Hierarchical Particle Swarm Optimisation [1]

In this section, we present a new adaptive approach to multi-view markerless articulated human body pose estimation from multi-view video sequences, using Particle Swarm Optimisation (PSO). We address the computational complexity of the HPSO, which successfully estimated a wide range of different motion with a fixed set of parameters, but incurred an unnecemetersssary overhead in computational complexity, as explained in Section 3.9.1.

When the range of motion we want to estimate with the same parameter settings is very wide, for example, from a simple slow walk to a fast karate kick, the easy solution is to set the starting inertia value $A$ high enough to guarantee that the exploration (rather than exploitation) is given sufficient priority and therefore the fastest motion will be estimated reliably. While the high inertia value is indeed necessary for sequences with fast and sudden motion, it is excessive in sequences where the subject is only walking. In such slow sequences, the high starting inertia value introduces an unnecessary computational overhead. To address this inconsistency, we formulate an adaptive extension of the HPSO approach, the APSO, where the starting inertia value, $A$, is adjusted on a frame-by-frame basis. Our adaptive approach, called APSO, preserves the black-box property of the HPSO in that it requires no parameter value input from the user. Instead, it adaptively changes the value of the search parameters online, depending on the quality of the pose estimate in the preceding frame of the sequence. We experimentally compare our adaptive approach with HPSO on four different video sequences and show that the computational complexity can be reduced without reduction in accuracy.

[1] Dr. Spela Ivekovic was the primary researcher for this work, while the author of the thesis was a contributing researcher. The contributions of the author include: implementiation of the algorithm, experimentation, and design of the scheme to set $\tau_0$ and $\tau_1$.

### 3.8.1 APSO Algorithm

In order to adjust the value of $A$ automatically, the adjustment process must exploit the information about the search performance in the preceding frame. The APSO approach therefore adaptively changes the next-frame starting inertia value for every hierarchical step in Table 3.2 by making use of two quality thresholds, $\tau_0$ and $\tau_1$: when the pose estimate $P_s^e(t)$ for a hierarchical step $s$ in the current frame is evaluated as good, $f(\mathbf{x}_s^e(t)) \geq \tau_1$, where $f$ is the fitness function, the search region in the next frame is kept small ($A_s^{t+1} = w_0$) as the tracker is thought to be on the target; when the pose estimate is very bad, $f(P_s^e(t)) < \tau_0$, the tracker is losing the target and hence the search region in the next frame is expanded significantly ($A_s^{t+1} = w_2$). When the estimate is average, $\tau_0 \leq f(P_s^e(t)) < \tau_1$, the search region is expanded moderately ($A_s^{t+1} = w_1$), where $w_0 < w_1 < w_2$. The process of adaptively changing the inertia value is illustrated with a state transition diagram in Figure 3.8.

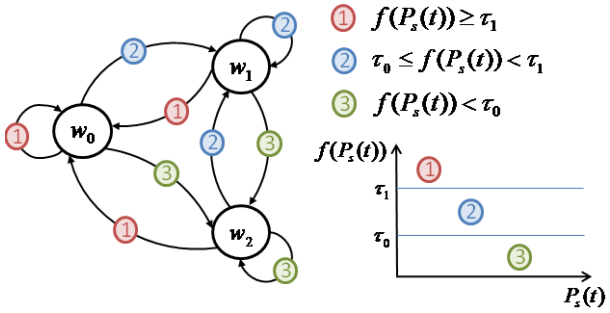



Figure 3.8: Adaptive inertia state transition diagram for step $s$ in the hierarchy. At the end of the search, the best pose estimate $P_s(t)$ is evaluated against two cost function thresholds, $\tau_0$ and $\tau_1$. The higher the $f(P_s(t))$, the better the pose estimate and the smaller the starting inertia for this hierarchical step in the next frame (the smaller the region that will need to be searched to find the pose estimate in the next frame).

*Best Pose Estimate.* The adaptive inertia scheme is also used to force the search into finding the best possible pose estimate in every frame, as follows. If the quality of the final pose estimate is bad or average, $f(P_s^e(t)) < \tau_1$, and the starting inertia value that was used is not the highest inertia value available,

$A_s^t = w_i, i < 2$, then the starting inertia value is increased to the next higher value, $A_s^t = w_{i+1}$, and the search is repeated. The process repeats until either the highest inertia value has been reached, $A_s^t = w_2$, or the pose estimate is sufficiently good, $f(P_s^e(t)) \geq \tau_1$. The value $A_s^{t+1}$ for the next frame is then determined as described in the previous paragraph and illustrated in Figure 3.8.

The rationale behind the use of this adaptive scheme is in the observation that even fast and sudden actions like, for example, karate kick (Figure 3.11), consist of segments with slow, medium and fast motion, and therefore searching with the highest inertia value in every frame would be excessive. The adaptive scheme favours a smaller inertia weight and as the experimental results in Section 3.9.3 demonstrate, this is not a bad assumption; the search time indeed decreases in comparison with HPSO without sacrificing the accuracy of the estimates. In fact, given the stochastic nature of the PSO, in our limited experimental evaluation the accuracy actually slightly increases owing to the search repeat strategy which corrects for bad starting values. Making the starting inertia value dependent on the quality of the pose estimate very effectively prevents the search from losing the target and ensures that even very erratic and sudden motion can be followed without diverging.

### 3.8.2 Setting $\tau_0$ and $\tau_1$

As a first attempt, we determined the values for $\tau_0$ and $\tau_1$ from a video sequence accompanied with ground truth optical motion capture data. The ground truth poses were used to evaluate the fitness function over a 200-frame sequence and the highest and lowest value of the fitness function were recorded. The interval between the highest and lowest value was then split into three equal bands and the boundaries of the middle band were used as $\tau_0$ and $\tau_1$. As we show with the experimental results, specifying $\tau_0$ and $\tau_1$ in this way does improve the efficiency of the pose estimation, however, we must stress that this is by no means the final

solution. Further research is necessary to find a principled way of setting these thresholds which will allow an optimal choice of the search region for every frame of the sequence.

## 3.9 Experimental Results

In this section, we evaluate the performance of our proposed human motion tracking using three different experimental setups. Firstly, we compare our human motion tracking system with comparable state-of-the-art tracking algorithms on the Lee walk and Surrey sequences, described below in Section 3.9.1. Secondly, we report a performance evaluation of our tracking system by varying the algorithm parameters. Finally, we compare the performance of APSO with HPSO using the and report our observations.

### 3.9.1 Comparative Experimental Tests

#### 3.9.1.1 Datasets and Algorithm Parameters

**Computational Framework.** To study the performance of the various tracking algorithms in the same conditions as much as possible, all tests were conducted using the Brown University framework. The various algorithms were plugged in, experiments run on the same data sets, and the same error measure calculated. Parameters specific to particular algorithms were set so as to optimise accuracy.

**Datasets.** We used four datasets: the *Lee walk* sequence included in the Brown University evaluation software and three sequences courtesy of the University of Surrey, UK [119] (*Jon walk*, *Tony kick* and *Tony punch*). The Lee walk dataset was captured with four synchronised grayscale cameras with resolution $640 \times 480$ at 60 Hz and came with the ground-truth articulated motion data acquired by a Vicon system, allowing for a quantitative comparison of the tracking results.

The three-dimensional error between the estimated and ground truth poses is the one implemented in the Brown University code, frequently used in the literature. The Surrey sequences were acquired by 10 synchronised colour cameras with resolution $720 \times 576$ at 25 Hz. No ground-truth data for the Surrey dataset is available; following Wang and Rehg [142], who used an overlap function to compare the results of various body tracking algorithms, we use the cost function values of the estimated poses as a means of comparison

**HPSO setup.** In all experiments, HPSO was run with only 10 particles. The PSO parameters (inertia weight model, stopping condition, search limits) and the covariance of the Gaussian distribution used for propagating the swarm into the next frame were kept the same across all the datasets. The starting inertia weight was set at two and the stopping inertia was fixed at 0.1 for all the sequences. This amounted to 60 PSO iterations per hierarchical step, with 12 hierarchical steps to yield 720 iterations in total. With 10 particles, it takes 7200 cost function evaluations per frame (one evaluation per iteration per particle) to estimate. Human biomechanical constraints (hard limits for rotation angles) are adopted as the search limits; such limits are also kept constant. The person size, including relative proportions among limbs, is established automatically from markers for the *Lee walk* sequence, and manually for the Surrey data set.

**APSO setup.** HPSO was run with only 10 particles; HPSO starting inertia weight was set to $A = 2$ and the stopping inertia was fixed at $w = 0.1$ for all sequences. This amounted to 60 PSO iterations per hierarchical step in HPSO or 7200 cost function (likelihood in PF) evaluations per frame. The APSO starting inertia values were set to $w_0 = 0.5$, $w_1 = 1.2$ and $w_2 = 2.0$, the stopping inertia was fixed at 0.1 and pose estimate accuracy thresholds $\tau_0, \tau_1$ were derived from the ground-truth pose estimates of the *Lee walk* sequence for every hierarchical step.

**PF/APF.** The Brown APF tracker reported in [8] uses a zero- velocity motion model: particles are diffused using a Gaussian distribution covariance, which is equal to the maximum inter-frame difference of joint angles and varies for every dataset. Unlike the original APF algorithm [30], for the *Lee walk* sequence the Brown software uses a hard prior trained from motion capture to initialise the tracking and eliminate particles with implausible poses. Obviously, this improves significantly the accuracy of APF tracking [8]. To ensure a fair comparison, we ran the particle filtering algorithms with biomechanical constraints as the hard prior, rather than action specific constraints. PF/APF were set up to use the same number of likelihood evaluations to find the solution. The reference number was provided by HPSO (7200 evaluations per frame, see above); we therefore ran the PF with 7200 particles, and the APF with 1440 particles and five annealing layers. We refer to this combination of tracking parameters as the canonical setup CS for PF and APF.

**PSAPF.** In addition to the above, we decompose the search space into 12 subspaces corresponding to the HPSO hierarchical steps described in Table 3.2. Bandouch et al. [10] combine the estimation of the root, torso, thighs and head into a single hierarchy, resulting in seven hierarchical partitions for the entire body pose estimation. However in order to ensure a fair comparison between HPSO and PSAPF, we modified their hierarchy to correspond to the hierarchical stages in HPSO. Finally the number of particles in PSAPF was also setup based on the number of likelihood evaluations per hierarchical step (600 evaluations). Thus PSAPF had 120 particles and five annealing layers or 7200 evaluations for 12 hierarchical steps (partitions). Finally, we refer this setup as the canonical setup CS.

**3D Error Measure.** In our experiments, we use the error measure adopted by Balan et al. [8] in their tracking software. The goodness-of-fit is obtained as a 3D error measure in millimetres, calculated as the average distance of 15 virtual

Table 3.3: The distance error calculated for the Lee Walk sequences and average time over 5 trials are reported.

| Sequence | LeeWalk 60 Hz | LeeWalk 30Hz | LeeWalk 20Hz |
|---|---|---|---|
| PF | 55.8±16mm<br>8hrs30min | 62.67± 19mm<br>4hrs15min | 101.3±25mm<br>3hrs10min |
| APF | 50.1±10.4mm<br>8hrs30min | 59.5± 12mm<br>4hrs15min | 94.1±21mm<br>3hrs10min |
| PSAPF | 48.1±12.8mm<br>5hrs | 54.95±12.1mm<br>2hrs50min | 89.59±23mm<br>2hrs |
| HPSO | 46.5±8.48mm<br>3hrs12min | 52.5±11.7mm<br>1hrs35min | 72.45±16.7mm<br>1hrs10min |

markers on the pose estimate with respect to 15 virtual markers derived from the ground truth pose. The 15 virtual markers are placed on the pelvis, neck, head, shoulders, elbow, wrists, hips, knees and ankles [8].

#### 3.9.1.2 Results

**Lee Walk.** HPSO performance compares favourably to the performance of PF, APF and PSAPF. Table 3.3 shows the error calculated as the distance between the ground-truth joint values and the values from the pose estimated in each frame, averaged over 5 trials. We also downsampled the sequence from 60 to 30 and 20 Hz to simulate faster motion. The Gaussian covariance for PF, APF and PSAPF was updated accordingly to optimise performance, while the covariance for HPSO was left unchanged. The distance error tabulated in Table 3.3 shows that HPSO performs comparably with APF, PF and PSAPF at the reduced frame rate ( 30 Hz) even with the unchanged covariance. However HPSO performs better than PF, APF and PSAPF at 20 Hz. Graphs comparing the distance error for 60 , 30 and 20 Hz sequences are shown in Figure 3.9, and visual illustrations of performance for the 20 Hz case in Figure 3.10 (a).

**Surrey sequences.** These sequences contain faster motion (punch, kick) than the *Lee walk* sequence; hence the covariance of the Gaussian distributions for PF,

Table 3.4: The cost function values of the estimated pose for the Surrey sequence. Smaller number means better performance.

| Sequence | JonWalk (5 trials) | Tony Kick (5 trials) | Tony Punch (5 trials) |
|---|---|---|---|
| PF | 0.37±0.03<br>4hrs55min | 0.6162±0.1183<br>3hr30min | 0.4995±0.11<br>3hr30min |
| APF | 0.334±0.03<br>4hrs55min | 0.465±0.03<br>3hr30min | 0.488±0.03<br>3hr30min |
| PSAPF | 0.332±0.025<br>3hrs45min | 0.45±0.02<br>2hr45min | 0.463±0.01<br>2hr45min |
| HPSO | 0.3046±0.0184<br>2hr20min | 0.3984±0.03<br>1hr30min | 0.40±0.22<br>1hr30min |

APF and PSAPF was again adapted accordingly to optimise performance, but HPSO's settings were left unchanged. For rapid and sudden motion in the *punch* and *kick sequence*, HPSO performed better than APF, PF and PSAPF (Figure 3.10(b) and 3.11) in terms of accuracy and stability of the tracker. The average overlap and standard deviation for a given sequence over 5 trials are shown in Table 3.15.

**Recovery from wrong estimates.** HPSO showed a systematic ability to recover from wrong estimates within a few frames; examples are shown in Figure 3.12 and Figure 3.14. PF and APF would, on occasion, lose track irrecoverably, i.e., the estimate would diverge. For example, in Figure 3.11, the right elbow is estimated wrongly by the APF and PF and never recovered. This behaviour was even more pronounced with PF. The success of HPSO at recovery behaviour is very likely due to the swarm behaviour which guarantees an exploration of a sufficiently wide region of the search space even with a limited number of particles.

**Automatic Pose Initialisation.** Finding the correct pose in the first frame is similar to recovering from wrong estimates, but the "previous" pose may be even further away. We used random starting positions of the skeleton model in the canonical pose (see Figure 3.13) as starting points for all algorithms. The starting skeleton (*canonical pose*) was visible from all cameras and oriented vertically,

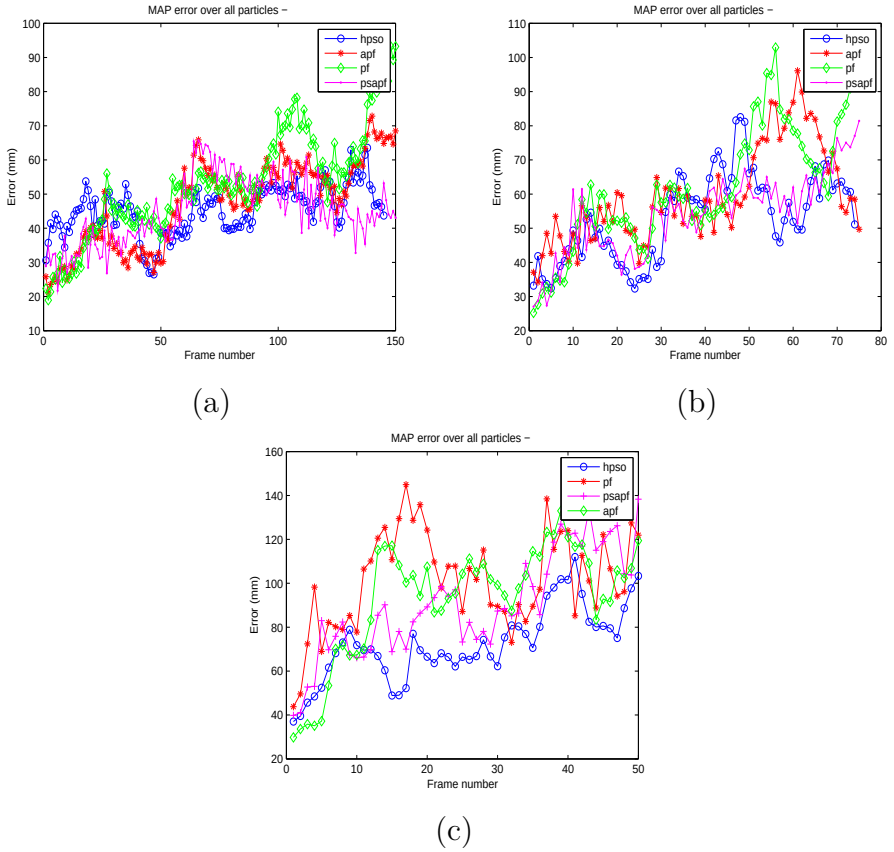
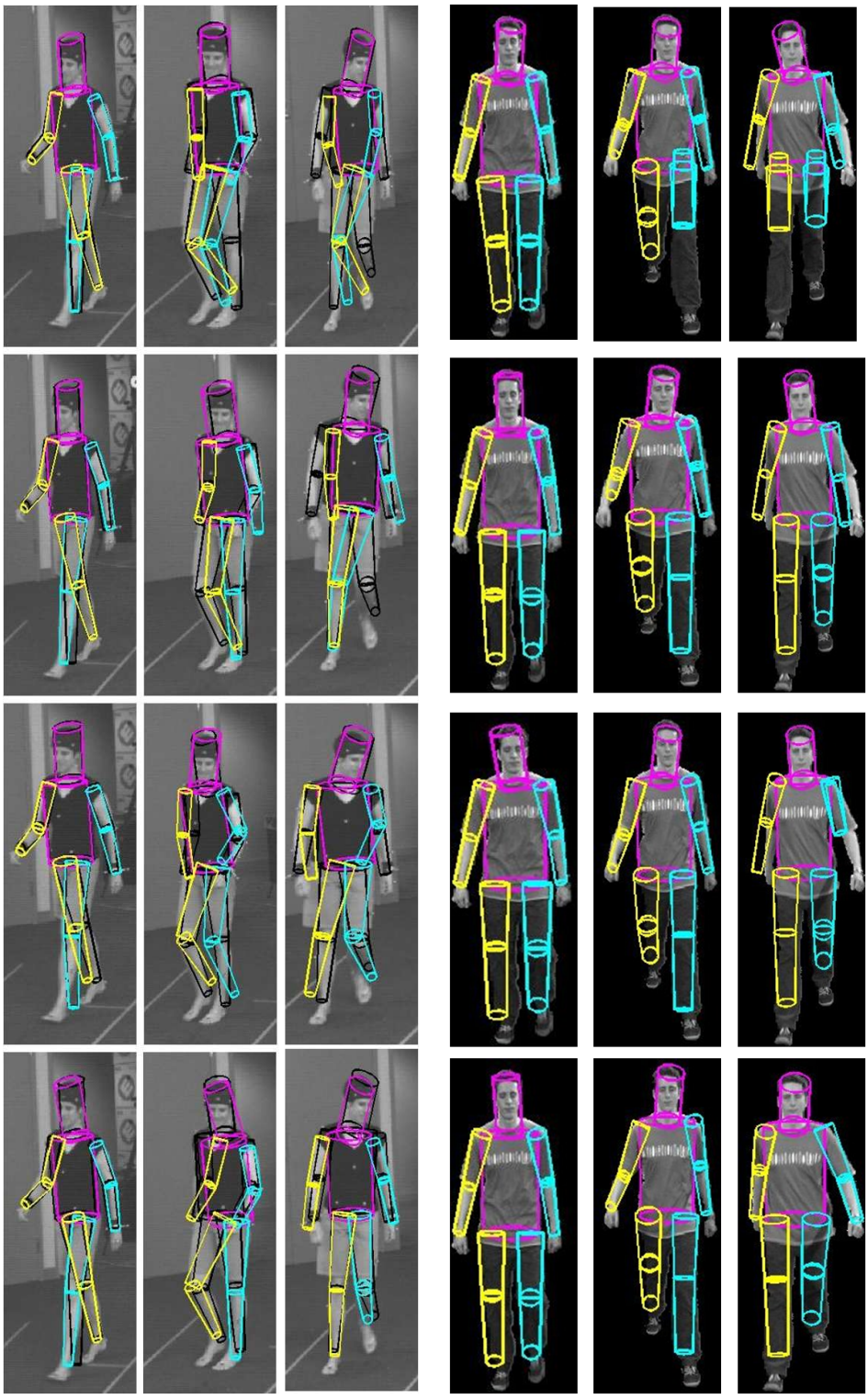


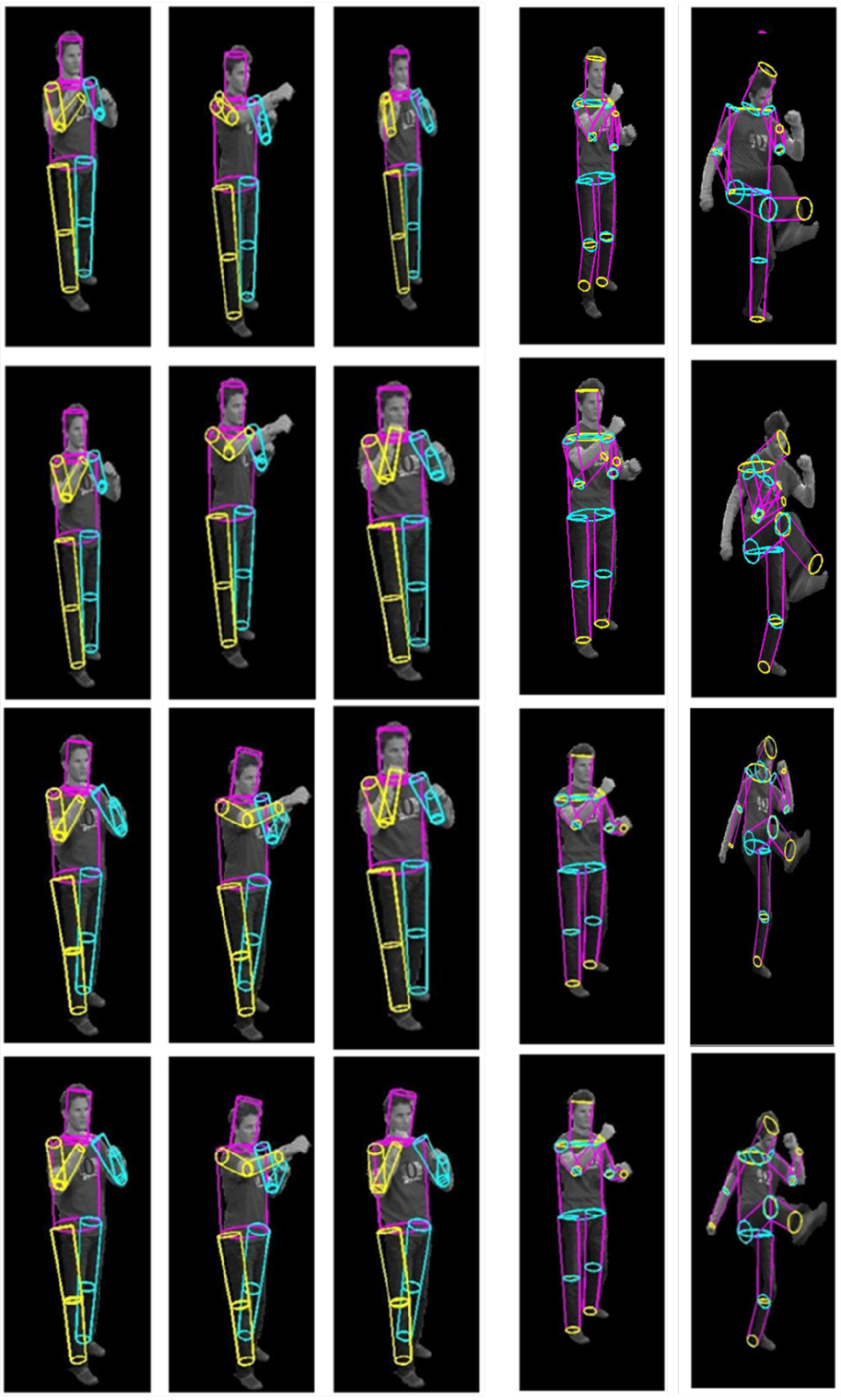

Figure 3.9: The distance error graph for (a) 60 Hz, (b) 30 Hz and (c) 20 Hz Lee Walk sequence.

two constraints satisfied by most sequences to be expected. We also set manually orientation in the direction of motion in the first frame. This is necessary because the canonical pose of the cylindrical body model of the Brown framework is symmetric with respect to the coronal plane, but the configuration of the reference frames in the joints is not. A more detailed model can eliminate the need for manual initialization, e.g., the SCAPE model adopted by Balan et al. [9]. We tested the automatic initialisation on all 4 test sequences. Initial canonical poses are shown in Figure 3.13(a,f). For the initial frame, the guiding cylinders are not used to provide temporal constraints, but only spatial constraints as described in Section 3.9.1.1. However as shown in Figure 3.13 (e,j), HPSO consistently found the correct position and orientation of the person in the initial frame, even without the guiding cylinder's temporal constraint. The particle filtering frameworks frequently failed to initialise automatically, as they expect a previous



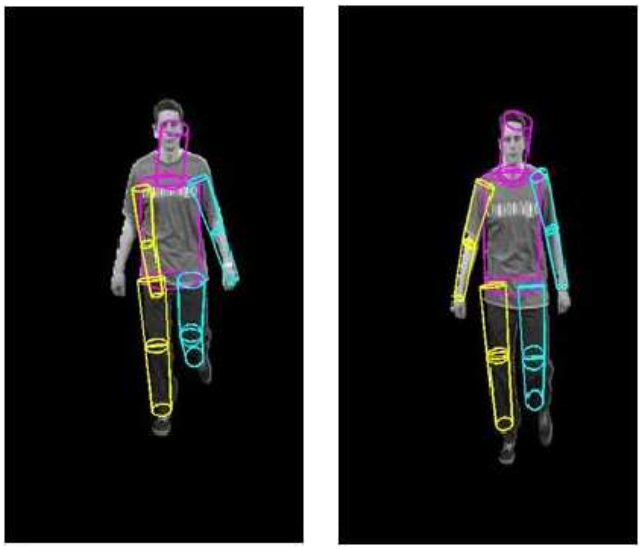
(a) (b)

Figure 3.10: The results of PF, APF, PSAPF and HPSO for the 20 Hz Lee Walk sequence (a) and Jon walk sequence (b) are illustrated in the first, second, third and last row, respectively. The black cylindrical body models (a) represent the ground-truth, while the coloured cylindrical body models (a,b) represent the estimated pose



(a) (b)

Figure 3.11: The results of PF, APF, PSAPF and HPSO for the (a) Tony Punch and (b) Tony Kick are illustrated in the first, second, third and last row, respectively.



(a) (b)

Figure 3.12: (a) An incorrect HPSO estimate (right arm); (b) the correct pose is recovered in the next frame.

estimate reasonably close to the current pose.

**Search Limits.** Search limits can be incorporated naturally and easily in PSO through simple checks on the particle positions. In particle filtering, instead, search limits are normally enforced through sample rejection and resampling. The samples with joint angles exceeding the search limits are discarded and sampling is repeated until the samples fall within the search limit. This process may increase the computational time by unpredictable amounts. Hence the search limits do increase the accuracy of the estimated pose, but at the cost of increased computational time. An experiment was conducted on the LeeWalk 60 Hz sequence to evaluate the benefits and shortcomings of incorporating the search limits. The performance of particle filtering algorithm using *CS* setup without any search limits was compared with the *CS* setup. As can be seen in Table 3.5, the accuracy increases significantly for all the particle filtering algorithms, however at the cost of significant increases in computational time. The times reported for all HPSO experiments here include biomechanical derived search limits.

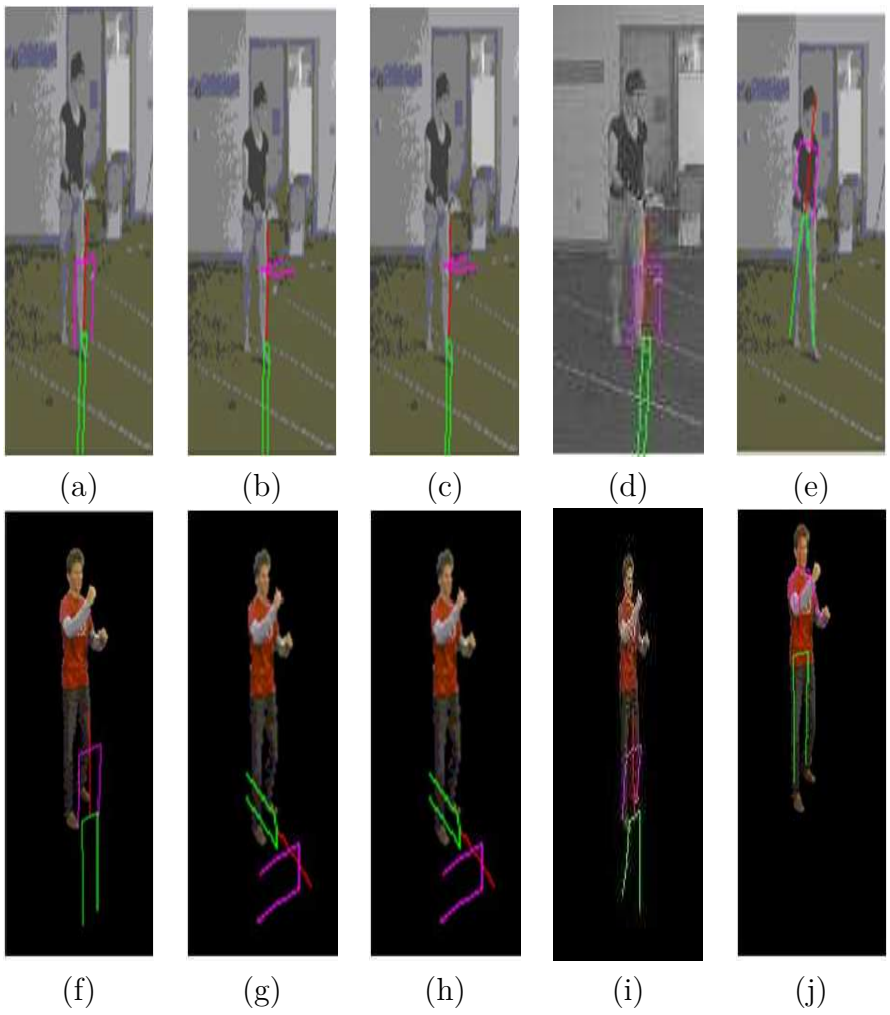



Figure 3.13: Automatic initialisation results for Lee walk (top) and Tony Kick (bottom) sequence. (a,f) The canonical initial pose for all three algorithms. (b,g) Unsuccessful PF , (c,h) unsuccessful APF and (d,i) unsuccessful PSAPF initialisation. (e,j) Successful HPSO initialisation.

Table 3.5: Distance errors and computation times with and without search limits for the Lee Walk sequence processed by the particle filtering algorithms.

| Sequence (LeeWalk 60Hz) | *CS* setup without Search Limits (5 trials) | *CS* setup (5 trials) |
|---|---|---|
| PF | 70.5±21.2mm 7hrs30min | 55.8±16mm 8hrs30min |
| APF | 68.38±17.5mm 7hrs30min | 50.1±10.4mm 8hrs30min |
| PSAPF | 63.8±19mm 4hrs23min | 48.1±12.8mm 5hrs |

**Time.** In generative tracking approaches, the time taken by an algorithm depends mostly on the number of likelihood evaluations; thus we used the same numbers of likelihood evaluations to compare the time taken by the different algorithms. For the same number of likelihood evaluations, the computation time for PF, APF and PSAPF is longer than tstill prone to the problemhat of HPSO. This can be attributed mostly to the implementation of the search limit constraints, which penalises particle filtering approaches but not HPSO.

The hierarchical optimisation scheme also reduces the computational complexity, since only selected cylinders corresponding to the body parts being optimised are projected for evaluation (Figure 3.18). In the Brown implementation, all cylinders are projected for PF and APF evaluation. In the case of PSAPF, the increase in time arises as a result of joint observation likelihood in the final partition, as described in Section 3.2.3. This issue is addressed in [67], under the condition of the observation likelihood being expressed as a product of sub-space likelihoods. Consequently, the partitioned sampling can be formulated by replacing the observation likelihood with an importance function, thus reducing the computational cost. However in the case of observations used by the Brown framework, i.e., silhouettes and edges, the likelihood observation cannot be factorised into a product of sub-space specific likelihoods, as a result of which, the hierarchical optimisation scheme could not be implemented.

**Accuracy.** The results in Table 3.3 and 3.15 suggest that HPSO is able to estimate the pose more accurately and consistently than PF, APF or PSAPF. On detailed observation of our results, we noticed that the performance of HPSO, PF, APF and PSAPF are nearly similar in the initial frames as shown in Figure 3.9. But the tracking performance of PF, APF and PSAPF greatly deteriotes as a result of divergence, unlike the performance of HPSO. The results in Table 3.3 are average 3D distance errors measured in *mm* over the entire sequence. A lower average distance error over the entire sequence (HPSO) not only demonstrates

better tracking performance but also a measure of divergence avoidance.

However HPSO is prone to occasional wrong estimates (e.g., Figure 3.12, a), which may depend on various factors, the relative importance of which is difficult to assess precisely but in specific, obvious input sequences: examples include noisy silhouette segmentation and self-occlusion creating ambiguous poses. We discuss this further in Section 3.9.2 along with the different approaches to address these issues. Examples of HPSO's prompt recovery from wrong pose estimates are shown in Figure 3.12 and Figure 3.14.

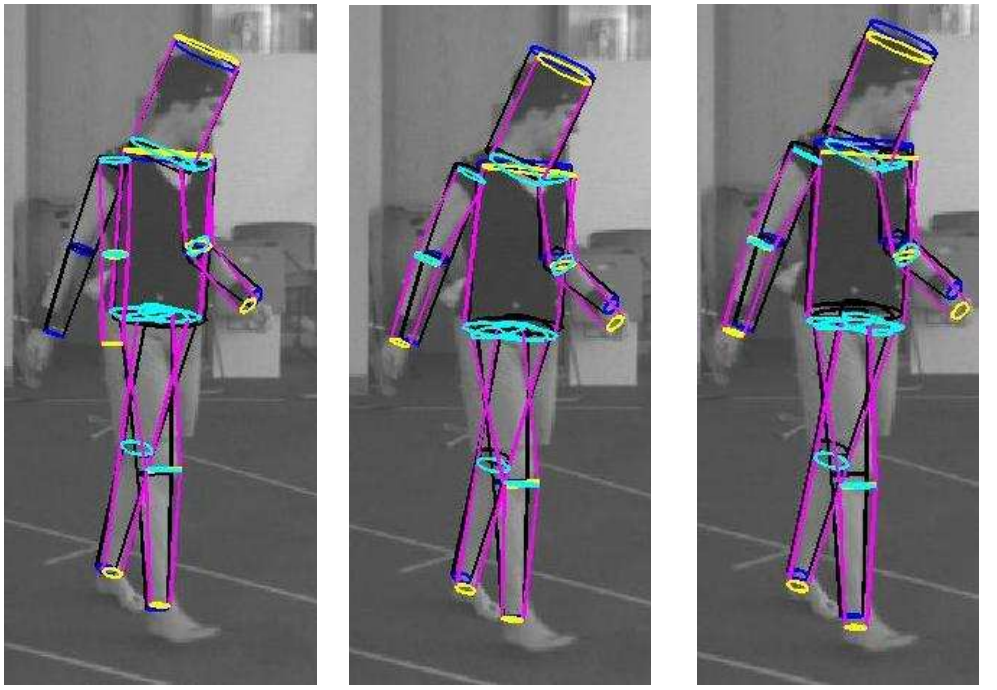
**Cost Function.** Balan et al [8] discuss the relative importance of edge and silhouette and conclude that the best tracking performance is obtained combining silhouettes and edges in the likelihood evaluation. Furthermore, when it comes to a single-feature likelihood evaluation (silhouette or edge), the silhouette-only likelihood evaluation is reported to perform better.

The model weighting function used in our experiments does not estimate how well the observed image features lie within the projected body pose. By using only a model weighting function, a wrong candidate pose can be assigned a high weight as seen in Figure 3.15(a) and Figure 3.16(a). In Figure 3.16(a) even though the right leg of the candidate body model is wrongly estimated, the body model has a high weight, as the right leg overlaps the left leg silhouette and edge.

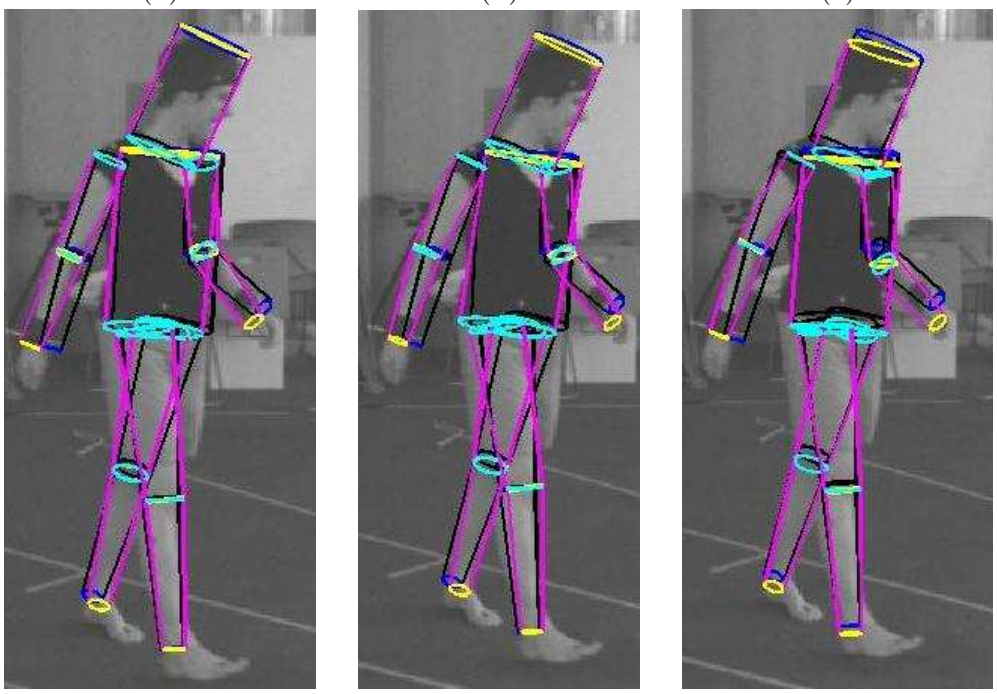
We address this problem by incorporating an additional *silhouette weighting* function, which accounts for silhouette pixels lying within the projected body pose. The silhouette weighting function $f(\mathbf{x}_n^i)$ of the $i$-th particle and $n$-th frame is given by:

$$S_n^i = \frac{M_n^i}{T_n} \tag{3.10}$$

where $T_n$ denotes the total number of silhouette pixels in the $n$-th frame and $M_n^i$

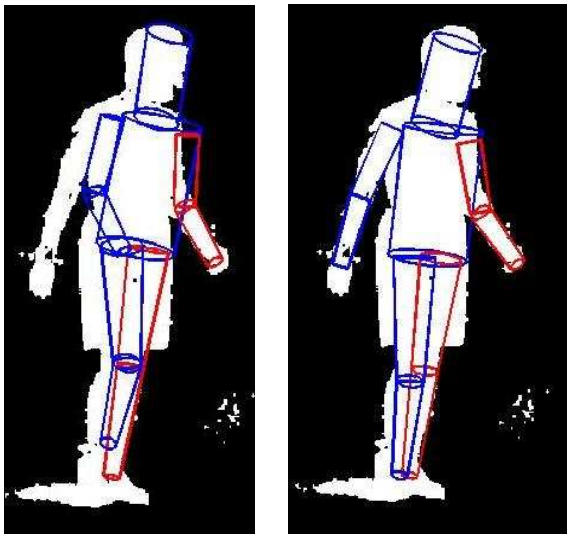


(a) (b) (c)



(d) (e) (f)

Figure 3.14: Results of Lee walk 20 Hz sequence illustrated for frames 13 (a,b and c) and 14 (d,e and f). The results of HPSO with 10, 20 and 50 particles are displayed in the first, second, and third column respectively. The first column (HPSO 10 particles) is an example of error propagation and recovery.

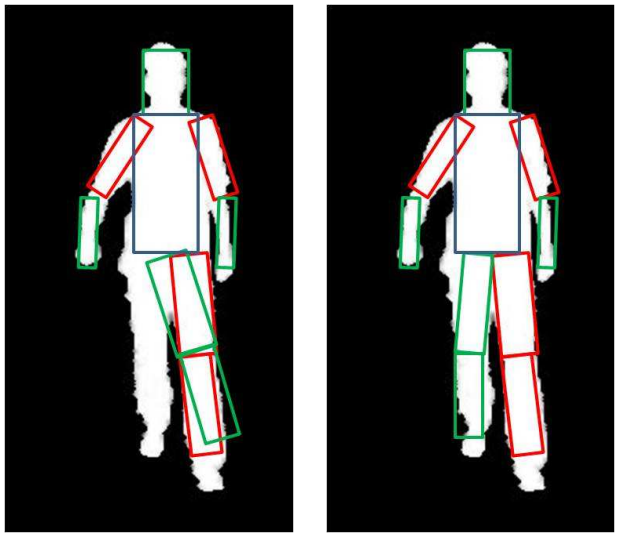


(a) (b)

Figure 3.15: Results of Lee walk 20 Hz sequence illustrated on frames 20 with different cost functions. The results of HPSO(10 particles) with a) model weighting function and b) combination weighting function are displayed.

represents the number of silhouette pixels lying within the projected body model corresponding to the $i$-th particle.

The silhouette weighting function is combined with the model weighting function to obtain a *combined weighting* function. We have evaluated the combined cost function (*CS* setup + silhouette weighting function) on the Lee walk 20 and 30 Hz sequence for the algorithms. The results obtained are compared with the model weighting function (*CS* setup) and tabulated in Table 3.6 and 3.7. Results suggest that the combined cost function does increase the accuracy of both the particle filtering algorithms and HPSO (though HPSO is more accurate), at the cost of an increase in computational time. The computational time increase is slightly worse for HPSO than for particle filtering algorithms. However the computational time for HPSO with the combined cost function is better than that of particle filtering algorithms. This is attributed to HPSO's hierarchical optimisation scheme (model weighting), as the computational time attributed to the silhouette weighting function is nearly similar for all the compared algorithms.



(a) (b)

Figure 3.16: An example of a) model weighting function and b) combined weighting function.

Table 3.6: The distance error calculated for the Lee Walk 20Hz sequences to evaluate different cost functions

| Sequence (LeeWalk 20Hz) | *CS* setup (5 trials) | *CS* setup with combined weighting function (5 trials) |
|---|---|---|
| PF | 101.3±25mm 3hrs10min | 88.44±24mm 3hrs45min |
| APF | 94.1±21mm 3hrs10min | 87.2±21mm 3hrs45min |
| PSAPF | 89.5±23mm 2hrs | 74.8±16.5mm 2hrs36min |
| HPSO | 72.45±16.7mm 1hrs4min | 68.7±11.6mm 1hr50min |

Table 3.7: The distance error calculated for the Lee Walk 30Hz sequences to evaluate different cost functions

| Sequence (LeeWalk 30Hz) | *CS* setup (5 trials) | *CS* setup with combined cost function (5 trials) |
|---|---|---|
| PF | 62.6±19mm 4hrs15min | 58.63 ± 17.71mm 5hrs20min |
| APF | 59.5±12.1mm 4hrs15min | 53.95 ± 10.15mm 5hrs20min |
| PSAPF | 54.95±12.1mm 2hrs50min | 51.81 ± 7.94mm 4hrs |
| HPSO | 52.5±11.7mm 1hr30min | 49.28 ± 14.41mm 2hrs30min |

### 3.9.2 HPSO Performance Evaluation against Parameter Changes

The quantitative results obtained in Section 3.9 suggest the reliable behaviour of HPSO with respect to the implementations of PF, APF and PSAPF available to us. We stress that HPSO was run throughout with an unchanged set of parameter values. In this final section, we investigate the effect of variations of the HPSO parameters on pose estimation accuracy. In particular, we vary the number of particles, number of camera views, compare the HPSO algorithm with the PSO algorithm to ascertain the benefits of the hierarchy, and evaluate the effect of the guiding cylinders.

**Number of Particles.** We varied the number of particles, $N$, within the canonical setup ($CS$) and evaluated the performance of HPSO. Unfortunately, the range of $N$ is limited by feasible computational times on our hardware. So we ran experiments with 10, 20 and 50 particles over 5 trials and results are tabulated in Table 3.8. Accuracy and consistency improve with an increase of $N$, as predictable, at the cost of increased computational time. HPSO with 20 and 50 particles is able to estimate the pose accurately and avoid error propagation as seen in Figure 3.14. However the number of likelihood evaluations per frame and computational cost increases with $N$: 20 particles result in $14,400$ likelihood evaluations and 50 particles in $31,600$ evaluations per frame. A full set of experiments to determine the value of $N$ after which no significant benefits occur was beyond our present hardware.

Additional tests were also run while studying cost functions, as shown in Table 3.8. There, HPSO was run with 20 and 50 particles using the combined cost function. As can be seen, the combined cost function does increase the accuracy of the pose estimation in addition to the improvement obtained by varying number of particles.

Table 3.8: HPSO's distance error in mm for the *LeeWalk* 20 Hz sequence with varying cost functions and varying number of particles

| Number of Particles | Model weight | Combined weight |
|---|---|---|
| HPSO (10 particles) | $72.45 \pm 16.7$ (*CS* setup) | $68.76 \pm 11.62$ |
| HPSO (20 particles) | $63.78 \pm 14.5$ | $55.73 \pm 12.7$ |
| HPSO (50 particles) | $58.76 \pm 14.3$ | $54.73 \pm 11.7$ |

Table 3.9: HPSO's distance error in mm for the *LeeWalk* 30 Hz sequence with varying number of cameras and *CS* setup

| Camera views | 4 cameras | 3 cameras | 2 cameras |
|---|---|---|---|
| HPSO | $52.45 \pm 11.7$ | 64.09±13.45 | 156.1±70.4 |

**Number of Camera Views.** In order to evaluate the performance of HPSO with fewer views, we ran an experiment using *CS* setup on *LeeWalk* 30 Hz sequence with 4,3 and 2 cameras and the results are tabulated in Table 3.9. Similarly we ran an experiment using *CS* setup on the *Tony Punch* sequence with 10, 8, 6 and 4 cameras and the results are tabulated in Table 3.10.

In the *LeeWalk* sequence, HPSO performs reasonably well with 3 cameras, but fails with 2. This is similar to the results by Balan et al. [8], where tracking fails with 2 cameras. Similarly, in the *Tony Punch* sequence, HPSO tracks reasonably well with 8, 6 and 4 cameras, without significant deterioration. Furthermore, HPSO tracking accuracy with 4 cameras is comparable to the performance of APF and PSAPF with 10 cameras.

**Hierarchical vs Non-Hierarchical PSO.** To evaluate the quantitative improvement brought about by hierarchical search, we ran an experiment using a non-hierarchical PSO search on the *Lee Walk* 20 Hz sequence. In order to ensure fair comparison, PSO setup was normalised to the number of likelihood evaluations of HPSO (7200). Thus for a 10 particle PSO, the number of inertia changes ($C$) was set to 720. The results (Table 3.11) show that the accuracy of PSO is

Table 3.10: HPSO's *Tony* punch sequence with varying number of cameras and *CS* setup

| Camera views | 10 cameras | 8 cameras | 6 cameras | 4 cameras |
|---|---|---|---|---|
| HPSO | 0.398±0.03 | 0.4077±0.03 | 0.4372±0.05 | 0.456±0.01 |

Table 3.11: PSO's performance on *Lee walk* 30Hz sequence compared with performance of PF,APF,PSAPF and HPSO taken from Table 3.5.

| | 10 particles ($C$=720) |
|---|---|
| PF | 62.67±19mm |
| APF | 59.5±12mm |
| PSO | 58.71±20.1mm |
| PSAPF | 54.95±12.1mm |
| HPSO | 52.5±11.7mm |

Table 3.12: HPSO's performance on *Lee walk* 30 Hz sequence with and without guiding cylinders

| LeeWalk 30 Hz | Guiding Cylinders | Without Guiding Cylinders |
|---|---|---|
| HPSO | 52.5±11.7mm | 103.4±23.2mm |

comparable to that of APF and PF, while the hierarchical approaches PSAPF and HPSO are better.

**Guiding Cylinders** . To evaluate the benefit of the guiding cylinders (henceforth GC) in the hierarchical optimisation scheme, we ran HPSO with *CS setup* on the *LeeWalk* 30 Hz sequence with and without GC. The latter involves projecting *only* the primary cylinders concerned with each hierarchical step. The results obtained are tabulated in Table 3.12; results show that GC bring a substantial increase in accuracy (about 50mm on this sequence). GC are mostly useful in recovery, when the limb to be estimated is obscured. For example, in Figure 3.17, where the right upper arm (primary cylinder) is obscured by the torso, the right lower arm (guiding cylinder) provides an effective constraint in finding the optimal pose.

**Error for Individual Body Parts.** HPSO error estimates for individual body parts (IBP)on *Lee walk* 30 Hz (CS setup) are reported in Table 3.13. The limbs are more prone to error, especially the lower arms and legs, whereas the head and pelvis are tracked fairly consistently. Our results reflect the particle filtering IBP error estimates observed in Balan et al. [8].

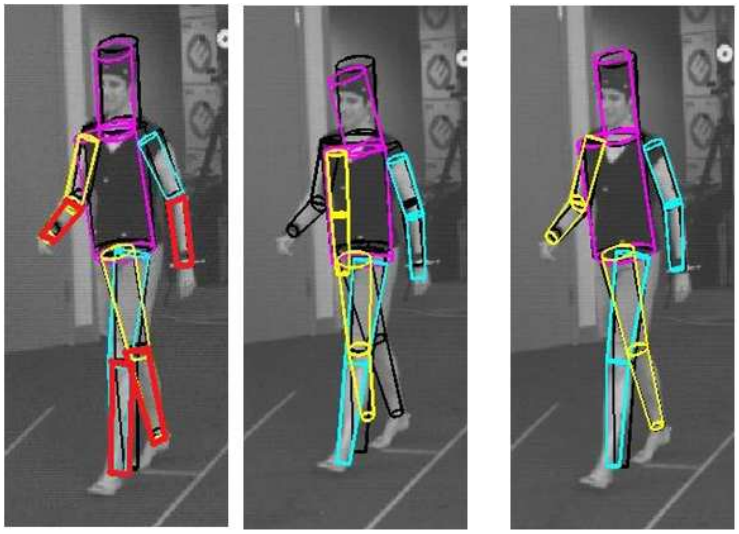


Figure 3.17: Lee Walk 30 Hz sequence results without (middle) and with (right) guiding cylinders for Frame 2. Left: the guiding cylinders (red cylinders) obtained from the previous-frame pose estimate (Frame 1) is shown. Middle: the right upper arm is obscured by torso and the lower arm is estimated incorrectly. Right: corrected pose recovered by HPSO with guiding cylinders.

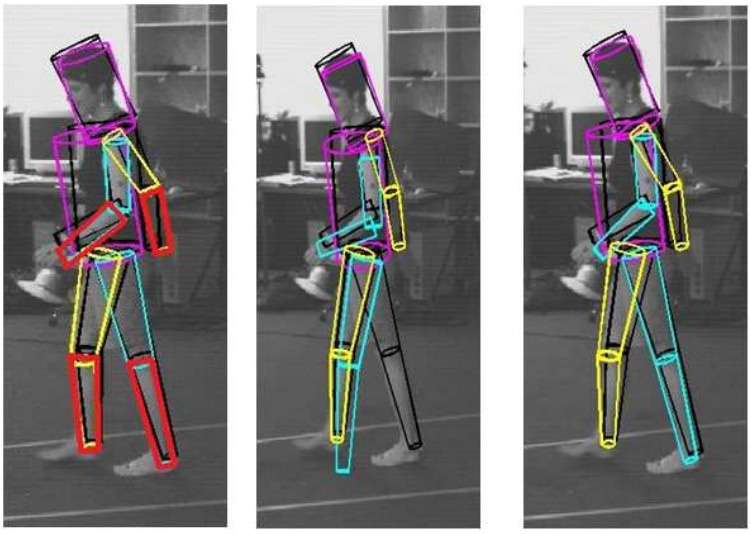


Figure 3.18: Lee Walk 30 Hz sequence results without (middle) and with (right) guiding cylinders for Frame 19. Left: the guiding cylinders (red cylinders) obtained from the previous-frame pose estimate (Frame 18) is shown. Middle: the left leg (thigh and knee) is inaccurately estimated. Right: the correct pose estimated by HPSO with guiding cylinders.

Table 3.13: HPSO error estimates for individual body parts on Lee walk @30 Hz (CS setup)

| Individual body parts | HPSO error estimate |
|---|---|
| Lower arms | 24.5 ± 9.8 mm |
| Upper arms | 14.28 ± 5.3 mm |
| Lower legs | 15.14 ± 5.3 mm |
| Upper legs | 11.8 ± 4.9 mm |
| Head | 6.5 ± 2.3 mm |
| Pelvis | 8.4 ± 2.9 mm |

### 3.9.3 Comparison of APSO vs HPSO

**Lee Walk Results.** We tested on a downsampled frame rate of 30 Hz instead of the original 60 Hz, to test with a faster action. The results in Table 3.14 show that APSO uses less time while also producing on average a more accurate pose estimate, most likely due to the search restarting several times in the same frame, every time with a better starting approximation, eventually producing a more accurate pose estimate. Table 3.14 shows the error calculated as the distance between the ground-truth joint values and the values from the pose estimated in each frame, averaged over 5 trials. HPSO took 70 sec per frame, while APSO varied between 40 sec and 100 sec per frame.

**Surrey Results.** Surrey test sequences contain faster motion than the *Lee walk* sequence. Again, our results for all tested sequences show that APSO reduces the tracking time while also producing more accurate pose estimates than all other approaches that we compared to. The average overlap and standard deviation for the Surrey sequence over 5 trials are shown in Table 3.15.

**Recovery.** An inherent problem associated with hierarchical search strategies such as HPSO is error propagation, whereby a wrong estimate in the initial subspace leads to an incorrect estimation in the subsequent subspaces as shown in Figure (3.19 a,b,c). However APSO inherently corrects for the error propagation with its search-restart strategy and thus produces fewer stray pose estimates Figure (3.19 d,e,f).

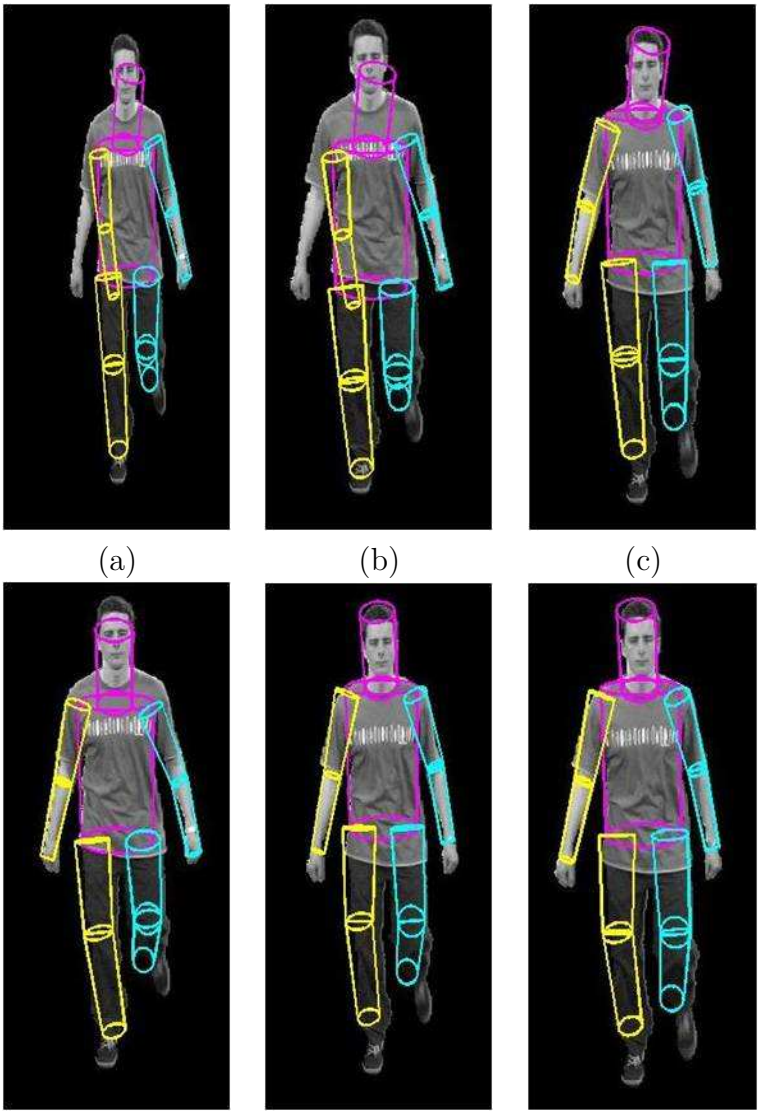


Figure 3.19: (a-c) an incorrect HPSO estimate due to error propagation is corrected within two frames. (d-f) APSO does not have the error propagation problem because of the adaptive inertia loop.

Table 3.14: Lee Walk sequence: the mean and standard deviation of the distance from the ground truth

| Sequence | HPSO<br>Mean ± Std.dev | APSO<br>Mean ± Std.dev |
|---|---|---|
| Lee Walk 30Hz<br>Average time taken (5trials) | 52.5±11.7mm<br>1 hr,35min | 50.8±10.4mm<br>1 hr, 5min |

Table 3.15: The cost function values of the estimated pose for the Surrey sequence. Smaller number means better performance.

| Sequence | HPSO<br>Mean ± Std.dev | APSO<br>Mean ± Std.dev |
|---|---|---|
| Jon Walk | 0.30±0.01 | 0.26±0.01 |
| Average time taken (5 trials) | 2hr,30min | 2hr,15min |
| Tony Kick | 0.39±0.03 | 0.38±0.03 |
| Average time taken (5 trials) | 1hr,30min | 1hr,15min |
| Tony Punch | 0.40±0.22 | 0.37±0.01 |
| Average time taken (5 trials) | 1hr,30min | 1hr,15min |

### 3.9.4 Discussion of Error Measures

In this section, we evaluate the performance of our proposed human motion tracking using errors in terms of average distance of 3D positions, which is the error measure calculated in the Brown University framework [8]. In addition to the Brown University framework, the HumanEva dataset [114], a popular markerless human tracking dataset, also adopt an error measure using an average distance of 3D positions. An error measure-based on joint angles variations is not usually considered as joint angle representations, potentially, give rise to multiple solutions for the same pose, complicating the error measure [8].

A distance-based error measure is an useful tool to evaluate tracking performance of different markerless human motion tracking systems with respect to marker-based motion capture systems, considered as state-of-the-art for biomedical and animation scenarios. Specifically, the error measure in terms of the 3D average distance over an entire video sequence, not only provides an accurate measure of an algorithm's tracking performance but also indirectly measures the algorithm's ability to avoid issues like divergence. An algorithm which estimates the pose successfully over the entire video sequence (HPSO), by avoiding divergence, would have a lower average distance error compared to an algorithm which loses track of the inital good estimate. Thus a low average distance error could be considered to be an useful measure to identify and evaluate algorithms for bio-medical and animation applications.

## 3.10 Conclusions and Future Work

This chapter has presented a hierarchical PSO algorithm (HPSO) and its adaptive version (APSO) for full-body articulated tracking using multiple synchronised views. PSO is applied to articulated body tracking, expanding on our previous work which applied PSO to *static* pose estimation. The quantitative results of our experiments show that HPSO with a small number of particles (10) yields results, under similar testing conditions, more accurate than those from the implementation of PF, APF, and PSAPF available to us. Advantages become particularly pronounced with fast and sudden motion (punch, kick). Unlike PF, APF, PSAPF and the local/global annealing approach, which rely on learning sequence-specific or weak (general) motion models, HPSO has demonstrated good performance without any motion prior. Our experiments, moreover, were conducted with the same algorithm parameter settings (e.g., inertia value) across all sequences used. Comparative results should be considered in this light.

HPSO successfully addresses the related problems of initialization and recovery. PF-related algorithms seem to depend often on external initialization. This is largely due to the effective communication between particles in the swarm search, which allows PSO to achieve results comparable with or better than those of the implementation of PF-based algorithms available to us, and of reported results of the local/global annealing approach. Successful initialization is achieved by simply running HPSO from the canonical model pose. In our experiments, tracking was always lost only temporarily and recovery achieved systematically after one or a few frames. Wrong pose estimates seem to depend mainly on poor silhouette segmentation in some cameras and the small number of particles used. We have ascertained experimentally that higher numbers of particles reduce positional errors. This number may depend on many factors (e.g., motion type, segmentation quality, number and positions of the cameras) and we have not investigated this point in detail as more powerful platforms than those used for

this study would be necessary. We notice that a body model composed only by cylinder, as the one borrowed here from [8] as a uniform basis for fair algorithm comparison, introduces a front-back ambiguity for poses in which all skeleton segments lie in a plane. This problem would be solved by nonsymmetric surface models, as used by Balan et al. [9]. The hierarchical, sequential structure of HPSO suggests that incorrect estimates at early stages of the kinematic chain will affect the accuracy of estimates for subsequent limbs. This problem is addressed by APSO, which additionally reduces the computational complexity and increases the accuracy. Moreover, nonlinear constraints created by the ranges of joint angles in the human body are incorporated naturally and very simply in the PSO paradigm.

Although APSO improves the tracking accuracy and addresses the problem of "error-propagation", it is still prone to wrong pose estimate in case of noisy and occluded silhouettes. Moreover, the computational cost for estimating the pose is high. In this regard, we propose to constrain the tracking problem, in the next chapter, by firstly, learning the low-dimensional subspace of common actions using charting, a non-linear dimensionality reduction algorithm. Prior models of pose and motion play an important role in 3D people tracking, addressing problems caused by occlusions, ambiguities and other noises. Secondly, to address the high computational cost, we set up the tracking formulation in the learnt subspace using a modified particle swarm optimisation and incorporate subspace hypothesis evaluation.

# Chapter 4

# Markerless Human Motion Tracking using Charting and Subspace Constrained PSO

## 4.1 Introduction

In Chapter 3, we presented a hierarchical PSO algorithm (HPSO) and its adaptive version (APSO) for full-body articulated human motion tracking using multiple synchronised views. We demonstrated good tracking accuracy on our experimental datasets, besides being able to automatically initialise, and recover from errors. But the HPSO tracking system does have drawbacks, arising from the high-dimensional search space, hypothesis evaluation method and absence of motion prior. Specifically, the high-dimensional search space combined with the expensive silhouette generation-based hypothesis evaluation results in a high computational cost. Additionally, the absence of motion prior makes HPSO highly dependent on the quality of the multi-view silhouettes. In this chapter, we propose a subspace-based full body tracking system, which aims to address

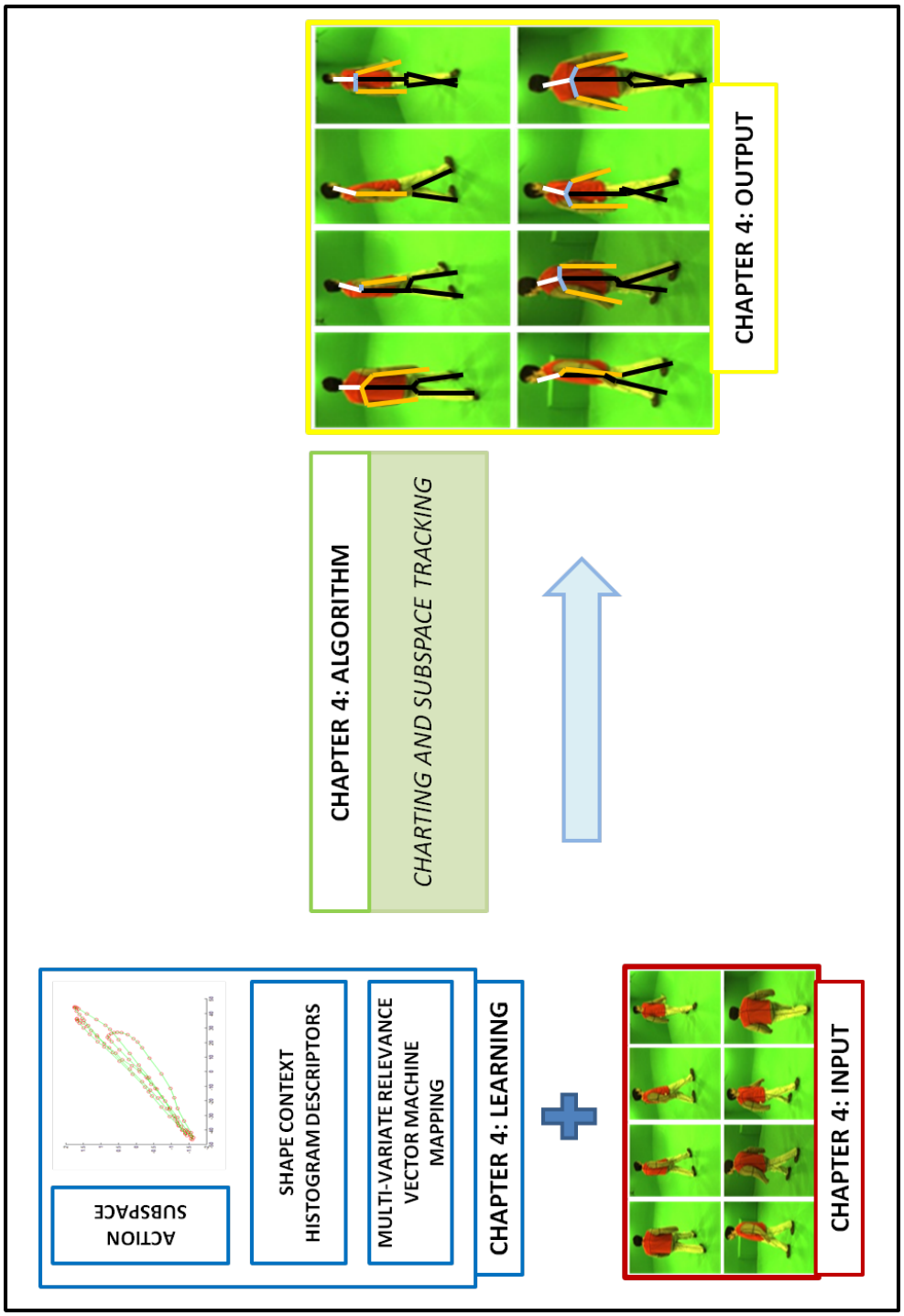



Figure 4.1: Overview of Chapter

the issues associated with HPSO.

*Context of Literature Classification.* In the context of our defined categorization of human motion tracking literature (Section 2.3.2), our second human motion analysis work presents a generative markerless multiple-view (**number of cameras**) human motion tracking algorithm using studio sequences (**acquisition environment**) and learnt action subspace (**extended motion model, as constraint**).

In context of three central ideas in generative subspace human motion tracking (Section 2.4.1), firstly, we use charting, a dimensionality reduction algorithm, to learn the action model. Secondly, we use a modified version of the particle swarm optimisation for subspace tracking. Thirdly, the hypothesis from PSO is evaluated without using any inverse mapping and in the subspace itself, which we explain in detail below.

*System overview.* In this chapter, we present a generative subspace tracking framework for markerless articulated human motion tracking using motion priors in multi-view sequences. We learn motion models of common actions in a low-dimensional subspace using *charting*, a nonlinear dimensionality reduction tool. Specifically, the articulated motion performed in a given sequence, captured by the evolution of the angles of a 31-dimensional 3D skeleton, is modelled in a low-dimensional subspace using *charting* [15]. Charting is a dimensionality reduction technique not yet used in human motion tracking, which estimates automatically the dimensionality of the embedded subspace and preserves closeness of similar poses in the subspace. Tracking takes place in the low-dimensional subspace. The generative component of our tracking system is a modified particle swarm optimisation (PSO) technique. In practice, we bias the swarm search to keep it close to the next pose isepredicted by the motion model, while allowing the particles to explore poses near the action subspaces, learnt during training. To evaluate a pose hypothesis, the silhouettes observed by the cameras are compared

with those generated by the candidate pose. The latter are obtained efficiently by mapping poses from the subspace to the space of silhouette descriptors. This mapping is learnt using a multivariate relevance vector machine (RVM) [125]. RVM sparsity contributes to the efficiency of pose evaluation. To our best knowledge, our work differs from the current literature in at least three ways. First, the use of charting to generate the subspace, has not been reported before for articulated body tracking. Second, PSO has been reported for articulated tracking but never in a low-dimensional subspace. Third, we propose a PSO variation designed for tracking on a latent-space action model. Tracking results with walking, punching, posing and praying sequences acquired in our studio and HumanEva sequence [114], demonstrate the good accuracy and performance of our approach.

*Chapter Layout.* The rest of this chapter is organised as follows. Section 4.2 summarizes some of the subspace learning techniques used in subspace tracking. Additionally, we introduce charting and discuss its characteristics in context of other subspace learning techniques. Section 4.3 presents our tracking framework, including the learning phase. Section 5.5 presents experimental results of our proposed system on our studio and HumanEva sequences. Finally in Section 5.6 we summarizes our work and suggest future developments.

## 4.2 Subspace Learning

Subspace learning is an important task in computer vision, which is based on the intuition that data lies on or near a complex low-dimensional subspace that is embedded in the high-dimensional space. Generally in machine learning literature and in various computer vision applications, the intuition of the presence of an embedded subspace in high-dimensional space is exploited to reduce the dimensionality of the high-dimensional data. Dimensionality reduction is the trans-

formation of high-dimensional data into a reduced dimensionality representation, as a good approximation of the original high-dimensional data. Dimensionality reduction plays an important role in many research areas, as it alleviates the curse of dimensionality, making the task of storing and searching data easier. A number of dimensionality reduction techniques have been proposed, varying in the method of obtaining the low-dimensional representation, they can primarily be classified as linear and non-linear techniques.

### 4.2.1 Problem Statement

The problem of dimensionality reduction can be defined as follows. Given a sequence of vectors, $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n]$, where $\mathbf{y}_n \epsilon \mathbb{R}^{\mathrm{D}}$ is a data present in a $D$-dimensional space, dimensionality reduction attempts to find a low-dimensional representation $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, where $\mathbf{x}_n \epsilon \mathbb{R}^{\mathrm{d}}$, and $d<D$ is the reduced dimension. Specifically, a mapping function $f(\mathbf{Y}) \rightarrow \mathbf{X}$ is learnt. In a few dimensionality reduction techniques [15, 59], the inverse mapping function $g(\mathbf{X}) \rightarrow \mathbf{Y}$ is also learnt.

### 4.2.2 Linear Subspace Methods

Principal components analysis (PCA) is a popular dimensionality reduction algorithm [135], which obtains a low-dimensional representation of the original data such that the maximum variance of high-dimensional data is preserved. This amounts to deriving the linear basis from the high-dimensional data set. The linear basis is of reduced dimensionality and encompasses the maximum variance in the data. The $d$ linear basis or principal components correspond to the eigenvectors/eigenvalue pairs calculated from the covariance matrix of the

high-dimensional data and are used to reduce the dimensionality of data.

Probabilistic variations of PCA also exist in literature. Tipping and Bishop [127] formulate such an approach to PCA known as *probabilistic PCA* (PPCA), where the high-dimensional data $\mathbf{Y}$ is represented as mapping from low-dimensional data $\mathbf{X}$ corrupted by Gaussian noise. This mapping is represented as follows,

$$\mathbf{Y} = \mathbf{W}\mathbf{X} + \varepsilon \tag{4.1}$$

where $\mathbf{W}$ is the mapping matrix which relates the subspace $\mathbf{X}$ to high-dimensional space $\mathbf{Y}$ and $\varepsilon$ is Gaussian noise. The goal of PPCA is to obtain the optimal mapping between $\mathbf{X}$ and $\mathbf{Y}$, which is achieviseed in four steps. First, a conditional probability model for the high-dimensional data is created expressing a linear relationship between $\mathbf{X}$ and $\mathbf{Y}$. Second, a Gaussian prior is specified over $\mathbf{X}$**.** Given the Gaussian prior and conditional probability, the subspace variables $\mathbf{X}$ are marginalised and an optimal $\mathbf{W}$ is solved using maximum likelihood estimation.

### 4.2.3 Non-linear Subspace Methods

In spite of its popularity, PCA and PPCA are limited to high-dim datasets lying on linear subspaces, thus they are not suitable for 3D full body human action, which typically lie on a non-linear subspace. Recently, several non-linear dimensionality reduction algorithms have been proposed, which are more suitable for learning non-linear subspaces. A few representative algorithms include local linear embedding (LLE) [101], Isomap [124], Laplacian Eigenmaps [11], Gaussian process latent variable (GPLVM) [59], local linear co-ordination (LLC) [102] and charting [15].

*Global Distance Preservation.* Isomap is a non-linear dimensionality reduction algorithm based on preserving the geodesic distances on the surface of the subspace represented as a graph. Given the geodesic-distance graph, Isomap attempts to find a lower dimensional space which preserves the geodesic or global distance on the subspace.

*Local Neighbourhood Structure Preservation.* While Isomap preserves the global distance, LLE and Laplacian Eigenmaps belong to a class of algorithm, which preserve local neighbourhood structure. In LLE [101], the local structure of high-dim data is represented as a linear combination of their nearest-neighbours, and LLE attempts to preserve the linear combination of nearest-neighbours in the low-dimensional subspace. In case of Laplacian Eigenmaps, the local structure is represented by the pairwise distances between nearest neighbors. Laplacian Eigenmaps, then, compute a low-dim subspace preserving the pairwise distance between nearest neighbours. The subspace distance preservation is done in a weighted manner, where distance between a given point and its closest neighbour contributes more to isedimensionality reduction cost function than neighbours which are farther away.

*Combination of Global Distance Preservation and Local Neighbourhood Distance Preservation.* The subspace learning techniques discussed so far obtain a non-linear low-dimensional space by either preserving global (Isomap) or local properties of the data (LLE, Laplacian Eigenmaps). A few techniques attempt to reduce the dimensionality by preserving both global and local properties. This is achieved by performing a global alignment of several locally linear models. LLC [102] and subspace charting belong to this class of subspace learning algorithms. In LLC, firstly, a mixture of locally linear models of high-dim data is obtained using Expectation Maximization (EM) algorithm, which are then aligned to obtain a single global low-dim subspace. In our work, we use charting to reduce the dimensionality, as explained in detail in Section 4.2.4.

Apart from the techniques discussed so far, the most popular and widely used technique in articulated human motion tracking is the *Gaussian process latent variable model* (GPLVM), a probabilistic non-linear dimensionality approach [59]. GPLVM is essentially a non-linear extension of PPCA, where Gaussian processes are used to map from the subspace $\mathbf{X}$ to the high-dimensional data space $\mathbf{Y}$ [60].

Though a non-linear extension, GPLVM differs from PPCA in several ways. Firstly, a Gaussian prior is defined over mapping function $\mathbf{W}$, instead of the subspace $\mathbf{X}$**.** Secondly, marginalising is done over $\mathbf{W}$ instead of $\mathbf{X}$. Finally, optimal $\mathbf{X}$ is obtained, instead of $\mathbf{W}$. Basically, GPLVM finds the optimal subspace $\mathbf{X}$ for a given $\mathbf{Y}$ (*optimal* $\mathbf{X}$), whereas PPCA finds optimal mapping from $\mathbf{X}$ to $\mathbf{Y}$ (*optimal* $\mathbf{W}$).

So far in this section, we have provided a brief overview of representative subspace learning algorithms belonging to different classes. We next describe in detail about charting, a subspace learning technique preserving global and local properties in a probabilistic framework. Charting forms an integral part of our subspace tracking and classification framework described in Section 4.3.

### 4.2.4 Charting

Charting constructs a nonlinear mapping from the high-dimensional space, $\mathbb{R}^{\mathrm{D}}$, to a low-dimensional subspace in $\mathbb{R}^{\mathrm{d}}$, where $d < D$. The mapping preserves local geometric relations in the subspace and is pseudo-invertible, so that the reverse mapping is also learnt. In charting similar poses in the high-dimensional joint angle space are mapped to the same region in the low-dimensional subspace, which is illustrated in Figure 4.2. The goal of charting is to estimate smooth, continuous mappings between the high-dimensional space and low-dimensional subspace. The mapping is expressed as a kernel-based mixture of linear projec-

tions.

The steps involved in charting include: (a) estimating directly the intrinsic dimensionality of the subspace from the training data; (b) obtaining locally linear, low-dimensional patches (*charts*), and merging them into a single low-dimensional space (*connection*); (c) computing the forward and reverse mappings between the high-dimensional and low-dimensional spaces. We next provide a description of all the steps, as described in [15]. Specifically, we provide of a brief overview of each step, in addition to the mathematical formulation and/or implementation, as described in [15].

#### 4.2.4.1 Estimating the Intrinsic Dimensionality and Local Scale

*Step Overview.* Given a $n$-frame sequence of $D$-dimensional joint angle vectors, $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{n}, \mathbf{y}_i \epsilon \mathbb{R}^{\mathrm{D}}$, the intrinsic dimensionality and the locally linear scale is estimated using a point-growth process, where a ball of radius or scale $r$, centered on each point, is grown with increased dimensionality and the number of data points contained in it, $n(r)$, is recorded. In order to estimate the locally linear scale, the point-growth process is based on the intuition that at some local scale, the subspace patch is locally linear with $d$ dimensions, so the number of data points $n(r)$ in the $r$-ball grows as $r^d$. The growth rate for a particular value of $r$ is tracked, $c(r) = \frac{d}{dlogn(r)} logr$ to estimate $d$. At lower scales noise will dominate resulting in *c(r)* being less than *1/d.* The same behaviour is also observed at non-linear scales, where a curvature of subspace patch occurs and *c(r)* is less than *1/d.* Thus the growth rate $c(r)$, which is maximum at the locally linear scale and is lower elsewhere.

*Implementation.* In practice, we follow the intrinsic dimensionality implementation, suggested by Brand [15], where an r-ball is expanded at every high-

dimensional data point and first peak in c(r), averaged over many neighbourhood data points, is obtained.

#### 4.2.4.2 Charting Step

*Step Overview.* An important step in the charting framework is the soft partitioning of the high-dimensional dataset into locally linear partitions by fitting a Gaussian mixture model (GMM) density to data. As the charting step is a precursor to the connection step (obtaining a single global co-ordinate system) the parameters of GMM are learnt with two constraints such that a) the data in each partition has minimal loss of variance between high and low-dim space, and b) neighboring charts should span maximally similar subspaces. The first criterion is obtained by fitting a GMM and maximising the likelihood. The second criterion is achieved by using a cross entropy-based prior, which ensures an alignment and overlap of neighbouring charts. The second criterion is important, as disagreement between neighbouring axis would lead to inconsistent projections of a high-dim data point resulting in uncertainities and distortions during low-dimensional data embedding (connection step).

Given the likelihood and prior, the posterior of GMM over the parameters is formulated, and MAP estimate is used to obtain the optimal parameters, which ensure that each chart span locally linear patches and neighbouring charts overlap and are aligned. Brand [15], has shown that under specific conditions the posterior would become unimodal and it can be maximised in closed form, which is described in the mathematical formulation below.

*Mathematical Formulation.* The first criterion of obtaining the minimal loss of variance is obtained by maximising the likelihood function of the GMM density given as

$$p(\mathbf{Y}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_j p(\mathbf{Y} \mid \mu_j, \Sigma_j) p_j = \sum_j \mathcal{N}(\mathbf{Y} \mid \mu_j, \Sigma_j) p_j$$

where each Gaussian component in the mixture, corresponds to a local neighbourhood centered around $\mu_j$ with axes defined by the eigenvectors of $\Sigma_j$.

The second criterion can be enforced through the cross-entropy between the Gaussian models of the two neighbourhoods, given as

$$D(\mathcal{N}_1 || \mathcal{N}_2) = \int dY \, \mathcal{N}(\mathbf{Y}; \mu_1, \Sigma_1) \frac{log \, \mathcal{N}(\mathbf{Y}; \mu_1, \Sigma_1)}{log \, \mathcal{N}(\mathbf{Y}; \mu_2, \Sigma_2)} \tag{4.2}$$

where the terms measure differences in size, orientation, and position, of neighbouring Gaussians with means $\mu_1$,$\mu_2$ and axes specified by the eigenvectors of $\Sigma_1$,$\Sigma_2$. The terms tend to zero when there is maximum overlap between the Gaussians. To maximize consistency between adjacent neighborhoods, and satisfy the second condition, cross-entropy is used within a prior, defined as

$$p(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = exp(-\sum_{i \neq j} m_i(\mu_j) D(\mathcal{N}_1 || \mathcal{N}_2)) \tag{4.3}$$

where $m_i(\mu_j)$ is a measure of co-locality and is defined as $m_i(\mu_j) \propto \mathcal{N}(\mu_j; \mu_{i,}, \sigma^2)$, with the scale parameter $\sigma$ specifying the expected size of a neighborhood on the subspace in sample space. A reasonable choice is $\sigma = r/2$, where $r$ is obtained from intrinsic dimensionality estimation. This choice of $\sigma$ introduces the locally linear scale into the GMM and ensures locally linear charts.

Given the likelihood and prior definition, which satisfy the two criterions, the posterior over the Gaussian parameters is then defined as

$$p(\boldsymbol{\mu}, \boldsymbol{\Sigma} | \, \mathbf{Y}) \propto p(\mathbf{Y} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{4.4}$$

Brand [15], has shown that under specific conditions the posterior would become

unimodal and it can be maximised in closed form. This is achieved as $\mu_i$ is fixed (GMM is centered on each high-dim data point), the $p_i$ is uniform, and neighboring charts span the same subspace, owing to the defined prior model. Given these specific conditions, the MAP estimates of the GMM covariances are:

$$\Sigma_i = \sum_j m_i(\mu_j)((\mathbf{y}_j - \mu_i)(\mathbf{y}_j - \mu_i)^T + (\mu_j - \mu_i)(\mu_j - \mu_i)^T + \Sigma_j)/\sum_j m_i(\mu_j) \quad (4.5)$$

The above equation is arranged in the form of fully constrained linear equations and solved for mutually optimal values. As each covariance $\Sigma_i$ is dependent on all other $\Sigma_j$, global information is bought into each local neighbourbood's description. Thus even if a given local subspace is noisy, the above formulation results in the corresponding local chart orienting itself as part of a globally optimal solution. Thus charting performs better than LLC when learning a smooth subspace with noisy data, as each chart MAP estimation depends on all other charts; bringing non-local information about subspace shape into the local description of each neighborhood[15].

#### 4.2.4.3 Connection Step

*Step Overview.* In the final step of charting, a connection for the set of local charts, specified by GMM, is obtained. Firstly, PCA is used in each chart, to reduce the dimensionality in each local chart. Specifically, a low-dimensional representation, $\mathbf{U}_k$, of the $k^{th}$ chart is obtained by PCA using the reference frame of the first $d$ eigenvectors of the chart's covariance matrix, $\Sigma_k$. Typically, several low-dimensional representations are obtained where all the high-dimensional data points are projected with respect to each PCA chart, $\mathbf{U}_k = \{\mathbf{u}_{ki}\}_{i=1}^n, \mathbf{u}_i \epsilon \mathbb{R}^{\mathrm{d}}$and $N$ is the number of data points. Given all the low-dimensional representations

$\mathbf{U}_k$, the goal of connection step is to sew together all charts into a single global low-dimensional subspace by using a weighted average projection.

To connect all charts in a single, consistent representation, each lower-dimensional chart, $\mathbf{U}_k$ is projected to the global low-dimensional co-ordinate system $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n}, \mathbf{x}_i \epsilon \mathbb{R}^{\mathrm{d}}$ using an: a) affine transform, say $G_k$ for the $k$-th chart and b) probability-based weighting $p_{k|\mathbf{y}}(y_i)$, which is the probability that $k-th$ chart generates point $y_i$.

An affine transform is used for the mapping, and it preserves collinearity of each locally linear chart. $G_k$ is obtained by solving a weighted least square problem. The weighted least square problem is based on the assumption, that if a data point point has non-zero probabilities in neighboring charts, then the neighboring affine transforms should map the data point to the same point in the global subspace.

Given the solved affine transform and the probability-based weighting function, the single global co-ordinate system is obtained, which is a smooth low-dimensional representation of the high-dimensional data. The solution for the affine transform is described in mathematical formulation given below.

*Mathematical Formulation.* The affine transform is solved by setting it up as weighted least square problem given as

$$\mathbf{G}^{\star} = [\mathbf{G}_1; \ldots; \mathbf{G}_k] = arg\,min_{\mathbf{G}_k;\mathbf{G}_j} \sum_i p_{y|k}(\mathbf{y}_i) p_{y|j}(\mathbf{y}_i) p_k p_j \left\| \mathbf{G}_k \left[\mathbf{u}_{ki}; 1\right] - \mathbf{G}_j \left[\mathbf{u}_{ji}; 1\right] \right\|_F^2 \tag{4.6}$$

This equation generates a set of homogeneous equations, from which solution for affine transform is calculated. Specifically, the solution for affine transform is obtained in the following steps: first, a constraint is added by anchoring the chart at origin $\mathbf{G}_1 = [I, 0]^T$; second, squared error of affine transforms is defined in terms of chart-to-origin chart inconsistency and chart-to-neighbouring chart

inconsistency as follows:

$$E = \sum_k \left[ \phi(\mathbf{G}\underline{\mathbf{U}}_k - [\mathbf{U}_1; \mathbf{0}])\mathbf{P}_k\mathbf{P}_1 + \sum_{j \neq k} \phi(\mathbf{G}\underline{\mathbf{U}}_j - \mathbf{G}\underline{\mathbf{U}}_k)\mathbf{P}_k\mathbf{P}_j \right] \tag{4.7}$$

where $\phi(\mathbf{X}) = trace(\mathbf{X}^T\mathbf{X})$ is the Gram trace. The first $\phi$ term in Eqn (4.7) penalises inconsistency with anchor chart, while the second term penalises pairwise inconsistency. $\underline{\mathbf{U}}_k = \mathbf{F}_k\,[\mathbf{U}_k; \mathbf{1}]$ ,where $\mathbf{F}_k = [\mathbf{0}, \ldots, \mathbf{0}, \mathbf{I}, \mathbf{0}, \ldots, \mathbf{0}]^T$is the indicator matrix, with identity matrix in *k-th* block. $\mathbf{P}_k = diag(p_{y|k}(\mathbf{y}_1), \ldots, p_{y|k}(\mathbf{y}_n))$ is the per-chart probability of all high-dim points.

In the next step, $\frac{dE}{d\mathbf{G}}$ is set to 0 and convex function is minimised and $\mathbf{G}$ is obtained as,

$$\mathbf{G}^\star = arg\,min_{\mathbf{G}}(trace(\mathbf{G}\mathbf{Q}\mathbf{Q}^T\mathbf{G}^T)) \tag{4.8}$$

where $\mathbf{Q}$=$\sum_{j \neq k}(\underline{\mathbf{U}}_j - \underline{\mathbf{U}}_k)\mathbf{P}_k\mathbf{P}_j$. $\mathbf{G}^\star$ is finally obtained by setting it to the eigenvectors associated with the smallest eigenvalues of $\mathbf{Q}\mathbf{Q}^T$. Once the affine transform is solved, each low-dimensional point, $\mathbf{x}_i$, is then obtained from the corresponding joint angle state vector, $\mathbf{y}_i$, as follows

$$\mathbf{x}_i \,|\, \mathbf{y}_i = \sum_k \mathbf{G}_k[\mathbf{u}_{ki}; 1]p(k \,|\, \mathbf{y})(\mathbf{y}_i) \tag{4.9}$$

which is the weighted average of the projections of sample $y_i$ into the low-dim space. The weighted average-based projection of high-dimensional datapoints implies that for a given point, the chart with higher probability has higher weight for the low-dimensional embedding, compared to chart which is not close and has a lower probability.

#### 4.2.4.4 Inverse Mapping

Unlike Isomap, LLE, the backward mapping from subspace to high-dimensional space is formulated within the charting framework itself. Given a probability density defined in the subspace, the surface passing through the weighted averages of the $\mu_i$ of all neighbourhood in which $y_i$ has non-zero probability is:ise

$$\mathbf{y}_i|\,\mathbf{x}_i = \sum_k p(k\,|\,\mathbf{x})(\mathbf{x}_i)(\mu_k + \mathbf{W}_k^T(\mathbf{G}_k[\mathbf{I};\mathbf{0}])^+(\mathbf{x}_i - \mathbf{G}_k[0;1])) \tag{4.10}$$

where $p_{k|\mathbf{x}}(\mathbf{x}_i)$ is the probability of $k-th$ chart generating $x_i$,$\mathbf{W}_k$ is the principal component analysis operator used in the connecting step and $(.)^+$ indicates pseudo-inverse.

The subspace learnt by charting is the integral element of our tracking system, which we describe in the section below, after providing a brief overview of the properties of charting.

#### 4.2.4.5 Discussion about Charting

*Charting vs GPLVM, LLC, PPCA.* Charting is a probabilistic subspace learning algorithm very closely related to the local coordination of global models method (LLC) [102], However, the latter model is prone to local minima, which is also case with algorithms like GPLVM and PPCA. Charting also performs better than LLC in the presence of locally linear noisy points. LLC in this scenario would estimate local partitions which are not smooth or continuous. Specifically, neighbouring charts are not constrained to be aligned resulting in scenarios, where neighbouring partitions could be perpendicular to each other. Charting, on the other hand, brings non-local information about subspace shape into the local description of each neighborhood, ensuring that adjoining neighborhoods

have similar covariances [15]. Thus even in the presence of locally linear noisy points, which are dense perpendicular to the subspace, the local chart are oriented parallel to the subspace as part of a globally optimal solution.

*Local Neighbourhood Preservation.* The learnt subspace representation was shown to preserve the geometry of high-dimensional local neighbourhoods in the subspace [15]. In charting, similar poses in the high-dimensional joint angle space are mapped to the same region in the low-dimensional subspace, which is illustrated in Figure 4.3 and Figure 4.2. This property arises as charting falls into the class of subspace learning algorithms, that perform a global alignment of locally linear model, preserving the local neighbourhood structure. The neighbourhood geometry preservation property can be used to model the subspace dynamics, as the periodic cycles of an action are well represented in the learnt low-dimensional subspace as shown in Figure 4.4 and Figure 4.5. Additionally the subspace for an aperiodic action is also illustrated in Figure 4.6. In our work, we exploit the neighbourhood and periodicity preservation property to propose a modified particle swarm optimisation for subspace tracking.

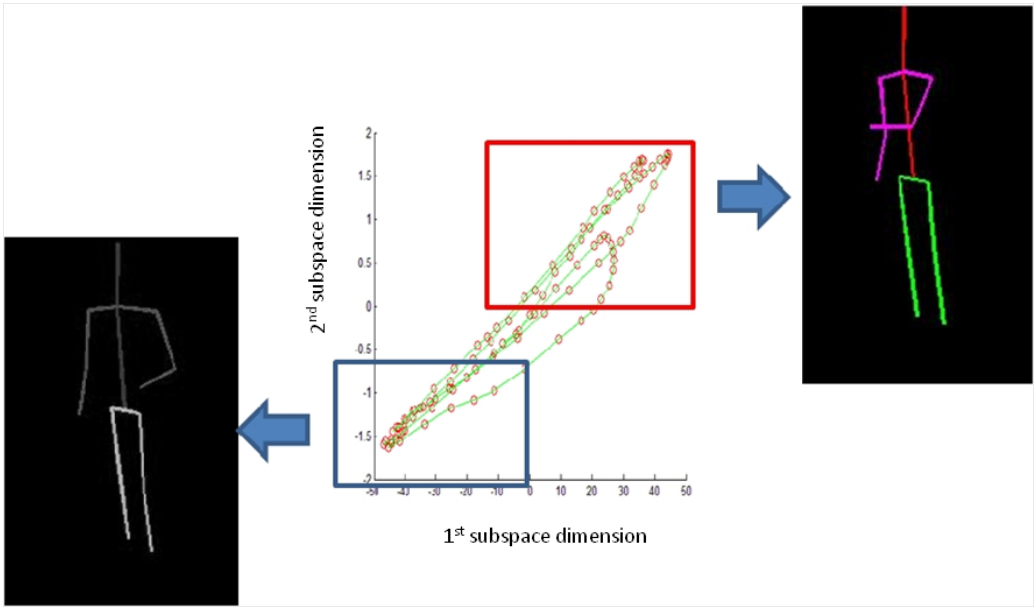



Figure 4.3: 2 dimensional action subspace for punch sequence, where similar high-dimensional joint angles are mapped to the same subspace region.

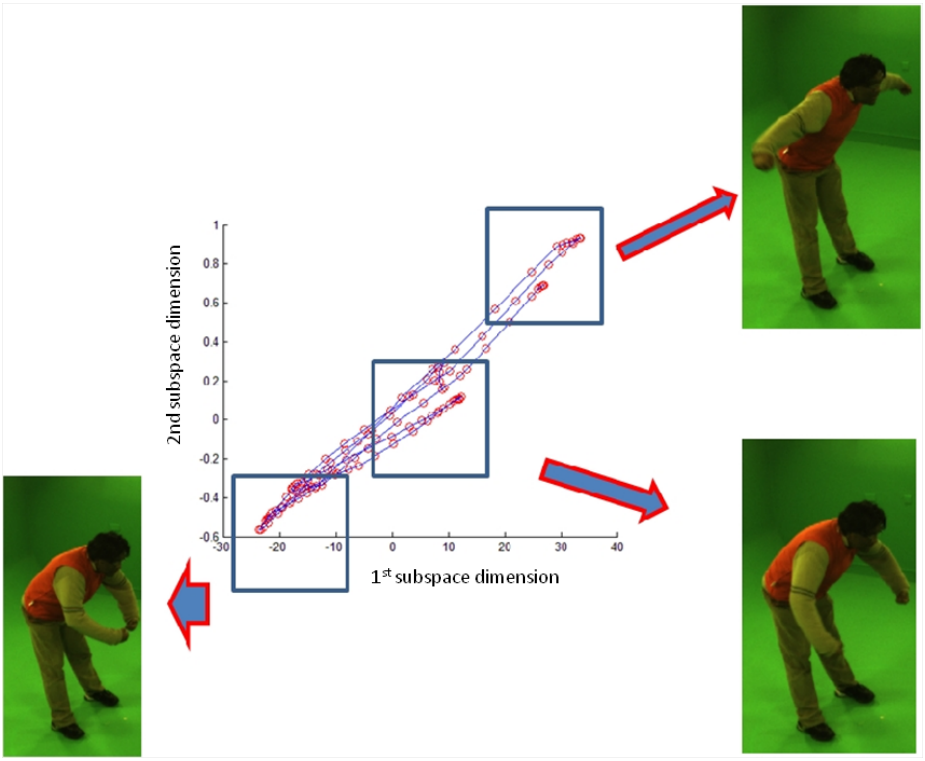



Figure 4.2: 2 dimensional action subspace for body posing sequence, where similar high-dimensional joint angles are mapped to the same subspace region.

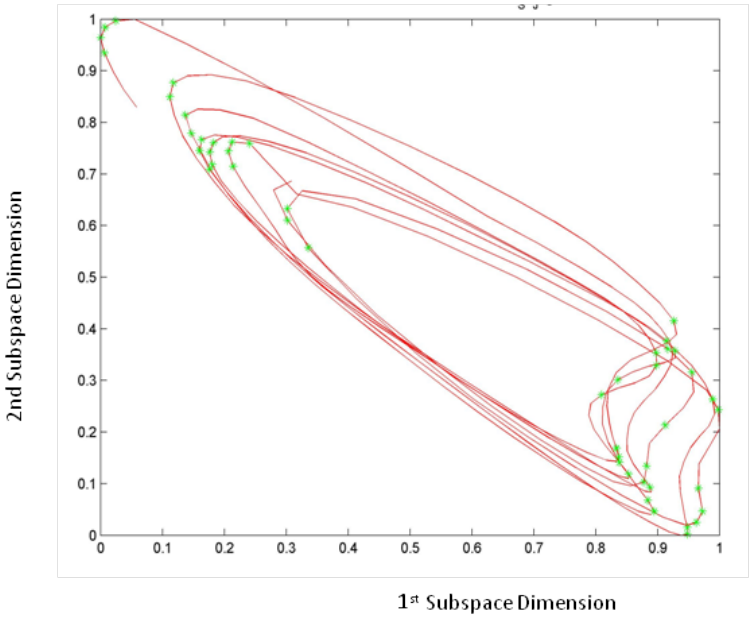



Figure 4.5: 2 dimensional action subspace for jog sequence. It can be observed that the subspace structure of the jog sequence is similar to subspace structure of the walk sequence, which is a similar action.

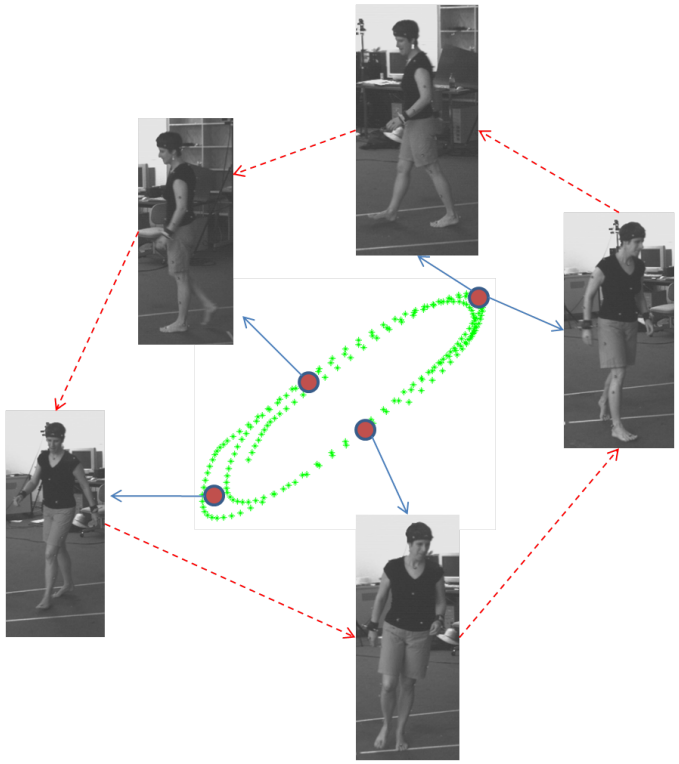


Figure 4.4: 2 dimensional action subspace for walk sequence, where similar high-dimensional joint angles are mapped to the same subspace region. Additional the periodic and cyclic nature of the action is well represented and approximated in the subspace.

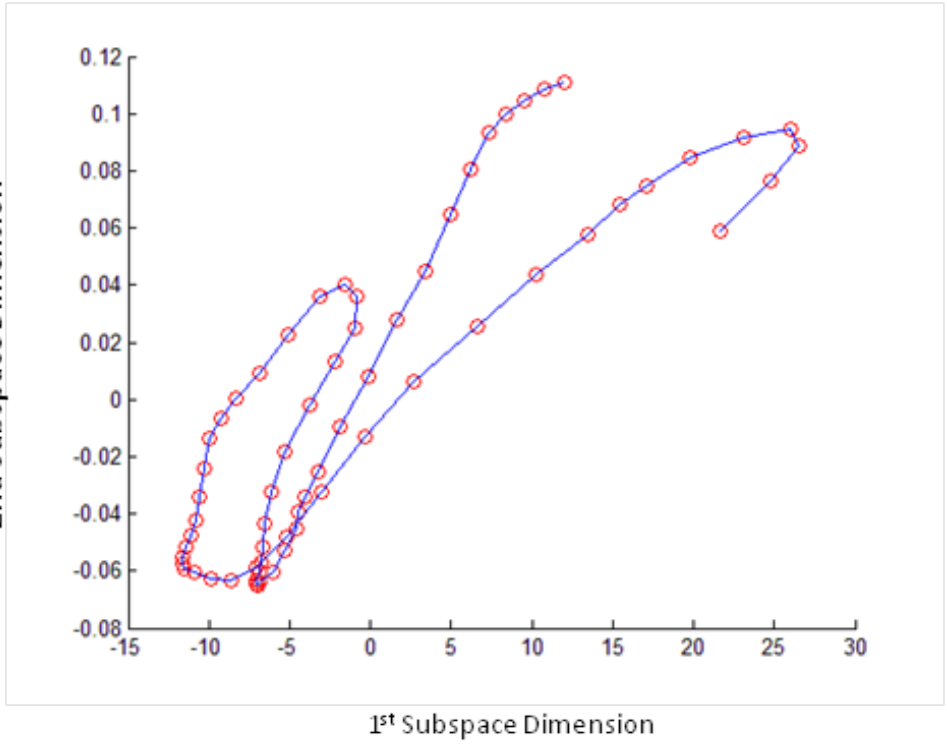



Figure 4.6: 2 dimensional action subspace for right football kick sequence, which was an aperiodic action, i.e. no cyclic sub-actions where observed.

## 4.3 Tracking Framework

*Motivation.* In this section, we present our subspace human motion tracking framework that explores the underlying low-dimensional subspace of the human body pose in common actions and accurately estimates the full-body human pose in multi-view sequences at reduced computational cost. Our tracking framework is formulated with two important requirements, namely: a) the exploitation of 2-dimensional action subspace to reduce the search space and increase the tracking performance; b) reduce the computational cost by avoiding the expensive silhouette generation-based hypothesis evaluation (Chapter 3). In this regard, our subspace multi-view human motion tracking system consists of two main phases, learning and tracking are shown in Figure 4.7 and 4.10. We first provide an overview of our tracking framework, before explaining the phases involved in greater detail.

*System Overview.* The learning phase is used to the: generate pose subspace (action model) using charting; generate a low-dim silhouette representation using vector quantized shape-context histograms (SCH) and finally learn the mapping from poses to silhouette descriptors using multi-variate relevance vector machines (MVRVM).

In the 2D tracking phase, a generative tracking algorithm is proposed using a modified PSO, designed to exploit the low-dimensional action representation, and effectively constraining the search. Moreover, the candidate poses are evaluated in the subspace itself by using the MVRVM mapping to SCH descriptor space. Additionally, we integrate the HPSO tracking algorithm introduced in Chapter 3 in our subspace tracking framework for the following functions: extraction of joint angles from our studio sequences; automatic initialisation of the subspace tracker; estimation of the root position and orientation; refining the subspace pose estimate. Please note that the root position and orientation step is neces-

sary as the pose estimated in the subspace corresponds to a 2D co-ordinate, and its inverse mapped pose corresponds to a 3D pose estimate without the root co-ordinates. As we employ HPSO to estimate the root, we also refine the complete full body pose using a local search, with fewer HPSO particles and few iterations. We next provide a detailed summary of the learning phase of our subspace tracking framework.

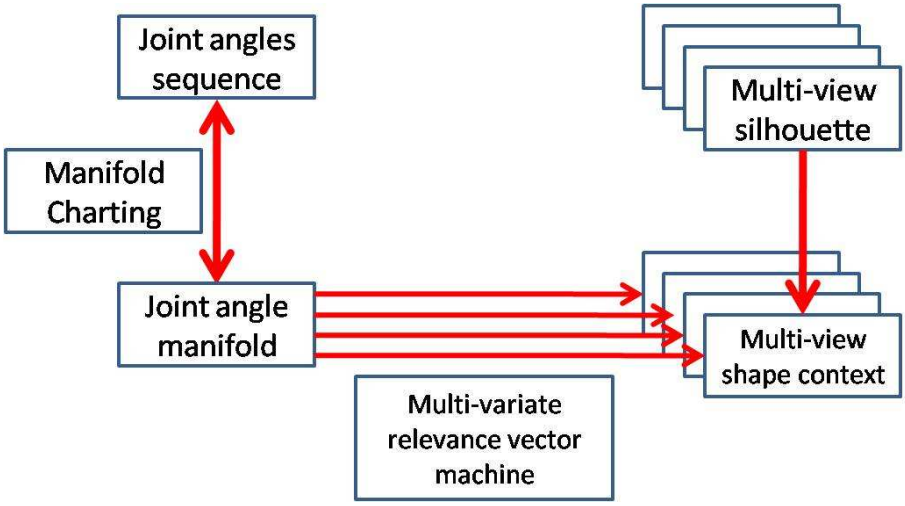



Figure 4.7: The learning phase of our system

### 4.3.1 Learning

*Overview.* The learning phase as shown in Figure 4.7 has four major steps. First, vectors of joint angles representing instantaneous body poses are estimated for all frames of training action sequences using either HPSO (studio sequences) or motion capture data in case of HumanEva dataset. The joint angles are then refined manually to avoid occasional significant errors, obtaining the training data for our system. Second, the low-dimensional subspace of the poses, $\mathbf{J}$, is obtained from the manually refined poses using charting [15]. Third, the multi-view image silhouettes are represented using multi-view shape context histograms [125] and reduced by vector quantization, yielding a set of shape decriptors, $\mathbf{S}$ for each

camera. Fourth and finally, we learn the mapping between **J** and **S** using multivariate RVM. An important property of subspace charting useful for subspace tracking is the preservation of high-dimensional data's local neighbourhood in the subspace also, which in practice implies the mapping of similar high-dimensional poses to similar regions in the subspace as shown in Figure 4.2 and Figure 4.3, this is an useful property for subspace tracking systems, as subspace dynamics can be easily modelled or exploited. We next provide a detailed description of the various steps involved in the learning phase of our subspace tracking system.

#### 4.3.1.1 Extracting Joint Angles and Learning the Subspace

For the first step in our learning algorithm, in case of our studio sequences, or sequences which do not have corresponding motion capture data, the joint angles are extracted using our HPSO algorithm explained in Chapter 3. Secondly, in order to obtain a smooth subspace, we smooth the extracted joint angles before learning the subspaces using charting. While learning the action-specific subspace, we omit the root position and orientation co-ordinates, as any action is independent of the root position and orientation. Moreover, the same action could have several root position and orientations making it difficult to model the action subspace. Additionally, as shown in Figure 4.8, we also smooth the joint angles before learning the subspace, as this results in an accurate subspace. Examples of the subspace learnt for different actions are shown in Figure 4.4, Figure 4.5, and Figure 4.6.

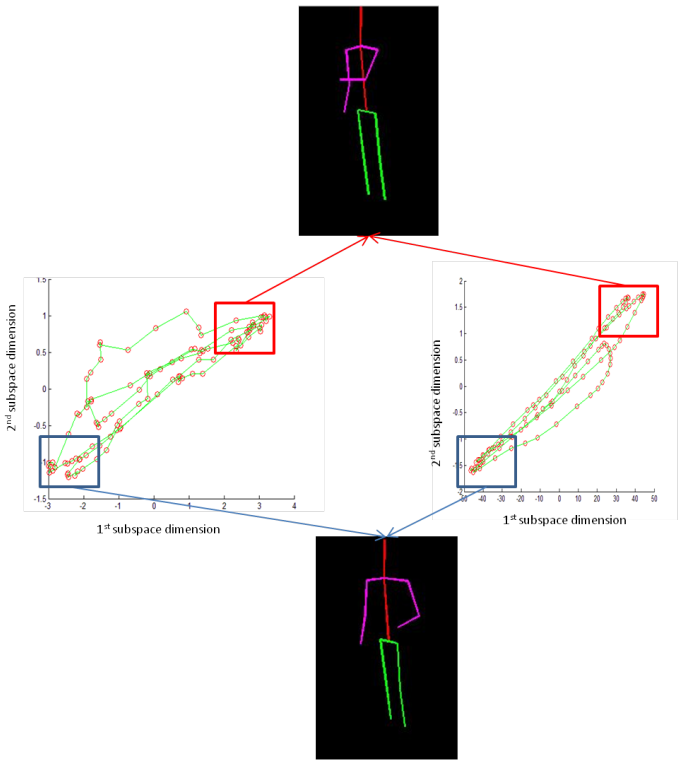



Figure 4.8: 2 dimensional action subspace for punch sequence, where smoothing of joint angles produces a smooth subspace (right) while original joint angles produce an unsmooth subspace (left).

#### 4.3.1.2 Low-Dimensional Representation of Silhouettes

Image features extracted from the video sequences are used for hypothesis evaluation in case of generative tracking systems and direct pose recovery in discriminative tracking systems. Typically, features such as edges, silhouettes, or colour are used. Amongst these features, silhouettes are widely used because they can be extracted relatively robustly. Furthermore, they are also insensitive to colour and texture variations in the human figure and most importantly they capture a great deal of information, useful for reasonably accurate 3D pose estimation. However, silhouettes are considered as high-dimensional image features, based on the image size, leading to increased computational cost in both generative and discriminative systems. This can be alleviated by using silhouette-based

descriptors, which encode the silhouette information at reduced dimension. An ideal silhouette descriptor should be a low-dimensional feature representation, that generalises over human figure variations, while simultaneously discriminating different body poses. In our subspace system, we use vector-quantised shape context histogram descriptors proposed by Agarwal et al. [3], after experimenting with Fourier descriptors. We first briefly explain about the Fourier descriptors and shape context descriptors, before explaining our motivation for choosing the shape context descriptor.

**Fourier Descriptors.** Fourier descriptors are shape descriptors, and the basic premise is to represent a silhouette or any shape by a fixed number $n$ of points $\{(d_1, e_1), \ldots, (d_n, e_n)\}$ on the boundary. The sampled points are then transformed into complex coordinates $\{z_1, \ldots, z_n\}$, which are subsequently transformed into the frequency domain using a Discrete Fourier Transform and obtaining the Fourier coefficients $\{f_1, \ldots, f_n\}$. Fourier coefficients with lower index represent the coarse shape information, with the first coefficient representing the position information. On the other hand, the finer details of the shape are encoded within the higher Fourier coefficients [91]. Position invariance of Fourier descriptors are obtained by setting the first Fourier coefficient to zero. Additionally, rotational invariance is obtained by ignoring the phase information. Finally the descriptors are made scale invariant by dividing the magnitude of all coefficients by the second coefficient $f_2$ [91].

**Shape Context Histogram Descriptors.** In our work, we use shape context histograms (SCH) to represent our multi-view silhouettes [3]. Shape context histogram are obtained from $n$ sampled points on the silhouette contours. A shape context histogram centered on a given contour point describes the spatial location of other *n-1* contour points in a histogram, where the histogram bins are uniform in log-polar space. The shape context histogram parameters are the number of radial bins, $\varphi$, and the number of log-distance bins $r$. Typically, the

default values are $\varphi = 12$ and r = 5 for the number of bins, resulting in a $n \times$ 60-dimensional SCH representation of the entire silhouette. Similar to Fourier descriptors, SCH is also translation, scale and rotation invariant. Translation or positional invariance is achieved automatically as only relative spatial information is measured, avoiding a global position information. In order to achieve scale invariance, distance between points are normalised by the mean distances between all point pairs. Finally, rotation invariance is obtained by setting the reference frame of shape context histogram to the tangent vector of each point, instead of positive x-axis [91].

Though SCH descriptors reduce the dimensionality of silhouettes, it is still high. Agarwal et al. [3] proposed the use of vector quantisation to further reduce the dimensionality of SCH descriptors by obtaining $m$ clusters from SCH space using K-means clustering over the entire video sequence. Each $n \times 60$-dimensional SCH descriptors are then represented in terms of $m$ clusters, using hard voting based on Euclidean distance. In our systems, for each camera, each instantaneous silhouette contour is sampled with 256 points, and the whole contour represented by 256, 60-dimensional SCH. In order to reduce the dimensionality of these descriptors, we vector-quantize the histograms using K-means clustering as described in [3], obtaining $40D$ silhouette descriptors. This representation is able to distinguish between different poses and have a degree of robustness to occlusion [91].

**Fourier Descriptor vs Shape Context Histogram Descriptors.** We compared Fourier descriptors and shape context histogram descriptors-based on their ability to generalises over human figure variations, while simultaneously discriminating different body poses. In order to evaluate the shape descriptors, we generate self-similarity distance matrices for shape descriptors (Fourier descriptor and shape context histogram descriptor) and qualitatively compare the generated matrices with self-similarity distance matrices of estimated human pose and

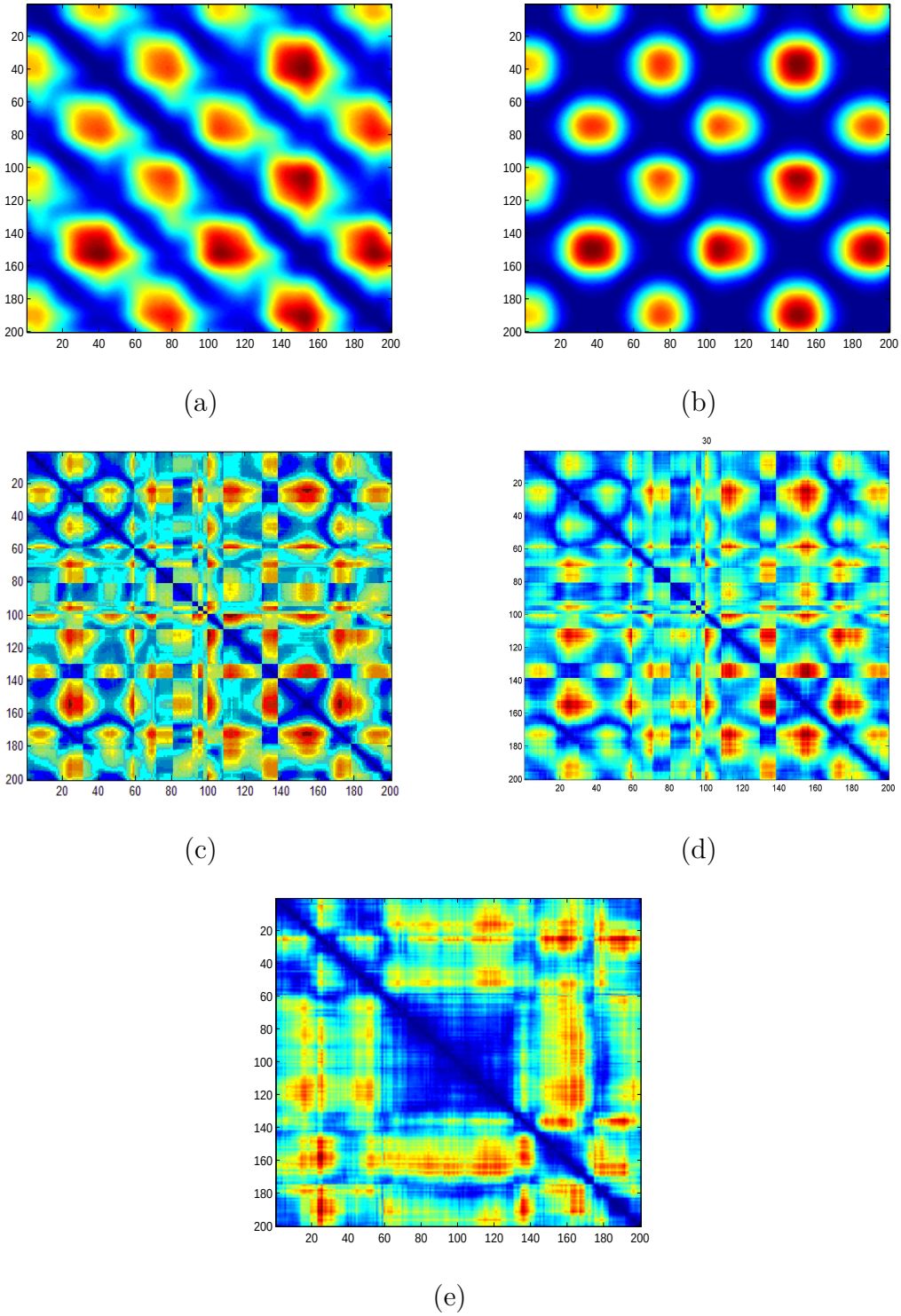



Figure 4.9: Lee walk Sequence: Self-similarity distance matrix for (a) High-dim joint angles; (b) Subspace of joint angles; (c) Silhouettes in one camera view; (d) Shape-context histogram in same camera view and (e) Fourier descriptors in same camera view.

silhouettes. On visual observation, as shown in Figure 4.9, we can observe that the self-similarity matrix obtained from SCH (Figure 4.9, (d)) is more similar to the human pose's distance matrix (Figure 4.9, (a)) and silhouette's distance matrix (Figure 4.9, (b)), whereas the self-similarity matrix of Fourier descriptors (Figure 4.9, (d)) does not capture the variation and similarities to the same degree. Based on this qualitative analysis, (visual observation) we represented our multi-view silhouette images using multi-view SCH descriptors.

#### 4.3.1.3 Mapping from Pose to Shape Descriptors

In this subsection, we explain about the mapping function used to learn the mapping from pose to shape descriptors. We begin with the relevance of the mapping function in our tracking framework, followed by an overview of the multi-variate relevance vector machine and finally, we provide a detailed overview of the mathematical formulation as described in [125].

*Relevance to Tracking Formulation.* To evaluate pose hypotheses efficiently, avoiding costly silhouette generation from 3D skeleton and surface models, we map poses from the subspace to shape descriptors which can be compared directly with the observed, or test, SCH representation. This mapping is learnt online using multivariate relevance vector machines (MVRVM).

*Overview.* MVRVM were introduced by Thayanathan et al. [125] as an extension of RVM for handling multivariate outputs. MVRVM is a sparse Bayesian regression technique used to find the optimal weights required for the mapping from a given input space to output space. In our case, the input space is the subspace pose space and the output space is the SCH representation. In MVRVM the optimal weights are obtained in two major steps; firstly, a posterior distribution over the weights, conditioned on a set of hyperparameters is obtained. Secondly,

given the posterior distribution, an optimal set of hyperparameters are found, which are used to derive the final optimal weights. During the training process, specifically in the second step, most hyperparameters tend to infinity, which result in the corresponding weights being set to zero. The remaining examples with non-zero weights are the relevance vectors. Relevance vectors are normally few, yielding a sparse representation, which in turn contributes to an efficient mapping. The rest of this section details the mathematical formulation of the MVRVM extension of the RVM framework, beginning with problem statement.

*Mathematical Formulation.* Given a set of training examples $\mathbf{V} = \{\mathbf{v}_n\}_{n=1}^{N}$consisting of input-output pairs of vectors $\mathbf{v}_n = \{\mathbf{r}_n, \mathbf{z}_n\}$, where $\mathbf{r}_n \epsilon \mathbb{R}^L$and $\mathbf{z}_n \epsilon \mathbb{R}^M$are the input and output vectors, the regression function is defined as

$$\mathbf{z} = \mathbf{C}\phi(\mathbf{r}) + \varepsilon \tag{4.11}$$

where $\phi(\mathbf{r}) \epsilon \mathbb{R}^{\mathcal{P}}$is set of basis function of the form $\phi(\mathbf{r}) = [1, \mathbf{G}(\mathbf{r}, \mathbf{r}_1), \ldots, \mathbf{G}(\mathbf{r}, \mathbf{r}_n)]$, $\mathbf{G}$ being a function comparing two feature sets, $\mathbf{C} = [c_1, \ldots, c_m]^T \epsilon \mathbb{R}^{M \times P}$ is the weight matrix containing the weights of the basis functions and $\varepsilon$ is the Gaussian noise, $\varepsilon \sim \mathcal{N}(\varepsilon; 0, \mathbf{S})$, where $\mathbf{S} = diag(\sigma_1^2, \ldots, \sigma_M^2)$ is the noise matrix. The goal of MVRVM is to find the optimal parameters $\{\mathbf{C}, \mathbf{S}\}$ of mapping function. MVRVM obtains the optimal parameters in two major steps, which we describe below.

*Posterior over Weights.* To obtain the optimal weights, firstly a posterior distribution over the weights conditioned on a set of hyperparameters are obtained. The posterior is obtained *by*, *f*irstly, defining a Gaussian prior over the weights of the basis function, conditioned on hyperparameters, is written as,

$$p(\mathbf{C} \,|\mathbf{A}) = \prod_{r=1}^{M} \mathcal{N}(\mathrm{c}; 0, \mathbf{A}) \tag{4.12}$$

where $\mathbf{A} = diag(\alpha_1^{-2}, \ldots, \alpha_p^{-2})$, where each element is the hyperparameter of the associated basis function. Then the prior is written as a product of separate Gaussians of each output dimension, After the prior has been defined, the likelihood distribution of the weight matrix $\mathbf{C}$ can be written as,

$$p(\{\mathbf{z}_n\}_{n=1}^N|\mathbf{C}, \mathbf{S}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{z}_\mathrm{n}|\mathbf{C}\phi(\mathbf{r}_\mathrm{n}), \mathbf{S}) \tag{4.13}$$

Finally, given the prior model and likelihood distribution, the posterior on $\mathbf{C}$ can be written as the product of separate Gaussians for the weight vector of each output dimension:

$$p(\mathbf{C}|\{\mathbf{z}_n\}_{n=1}^N, \mathbf{S}, \mathbf{A}) \propto p(\{\mathbf{z}_n\}_{n=1}^N|\mathbf{C}, \mathbf{S})p(\mathbf{C}|\mathbf{A}) \propto \prod_{r=1}^{M} \mathcal{N}(\mathrm{c_r}|\mu_\mathrm{r}, \Sigma_\mathrm{r}) \tag{4.14}$$

where $\mu_r = \sigma_r^{-2}\Sigma_r\Phi^T\tau_r$and $\Sigma_r = (\sigma_r^{-2}\Phi^T\Phi + \mathbf{A})^{-1}$are the mean and covariance of the distribution of $w_r$. $\tau_r$ is the vector with *r-th* component of all output vectors. $\Phi = [1, \phi(r_1), \phi(r_2), \ldots, \phi(r_N)]$ is the design matrix of all basis functions.

*Optimal Set of Hyperparameters.* Given the posterior over the weights, the optimal weight matrix is chosen, by obtaining the set of hyperparameters that maximise the likelihood in Eqn (4.14). In order to obtain the optimal hyperparameter set, the data likelihood is marginalised over the weights:

$$p(\{\mathbf{z}_n\}_{n=1}^N|\mathbf{A}, \mathbf{S}) = \int p(\{z\}_{n=1}^N|\mathbf{C}, \mathbf{S})p(\mathbf{C}|\mathbf{A})d\mathbf{C} \tag{4.15}$$

$$= \prod_{r=1}^{M} \int \mathcal{N}(\tau_\mathrm{r}|\mathrm{c_r}\Phi, \sigma_\mathrm{r}^2)\mathcal{N}(\mathrm{c_r}|0, \mathbf{A})$$

$$= \prod_{r=1}^{M} |\mathbf{H_r}|^{-\frac{1}{2}} exp(-\frac{1}{2}\tau_r^T\mathbf{H}_r^{-1}\tau_r),$$

where $\mathrm{H}_r{=}\sigma_r^2 I + \Phi\mathbf{A}^{-\mathbf{1}}\Phi^T$. An optimal set of hyperparameters $\{\alpha_j^{opt}\}_{j=1}^{P}$ and noise parameters $\{\sigma_r^{opt}\}_{r=1}^{M}$ is obtained by maximising the marginal likelihood.

*Optimal Weights.* The optimal hyperparameters are then used to obtain the optimal weight matrix:

$$\mathbf{A}^{opt} = diag(\alpha_1^{opt}, \ldots, \alpha_p^{opt}) \quad \Sigma_r^{opt} = ((\sigma_r^{opt})^{-2}\Phi^T\Phi + \mathbf{A}^{opt})^{-1}$$
$$\mu_r^{opt} = (\sigma_r^{opt})^{-2}\Sigma_r^{opt}\Phi^T\tau_r \qquad \mathbf{C}^{opt} = [\mu_1^{opt}, \ldots, \mu_M^{opt}]^T$$

### 4.3.2 Tracking

In the previous section, we provided a detailed overview of our learning framework, which was used to obtain the components required for our subspace tracking and evaluation, namely, charting-based action subspace, mapping from pose space to image space and SCH image representation. In this section, we explain in detail about the tracking component of our proposed subspace framework, with specific focus on subspace tracking and evaluation, after providing a brief overview of the tracking phase.

*Overview.* Tracking has three major components: automatic initialisation of 3D pose, pose estimation in subspace, and final pose refinement in joint angle space, including estimation of the root position in 31D. In the first component, HPSO described in the previous Chapter is used to initialise the high-dimensional pose, which is mapped to the learnt subspace and used to initialise our modified PSO. The modified PSO is used to obtain an optimal subspace pose estimate, which is a 2D co-ordinate. In the final component, the 2D subspace pose co-ordinate is inverse mapped to the higher dimensional space and the root position and orientation are obtained. Additionally, the 31D pose estimate is also refined using a local search. An overview of subspace tracking is shown in Figure 4.10.

Each step is described briefly below.

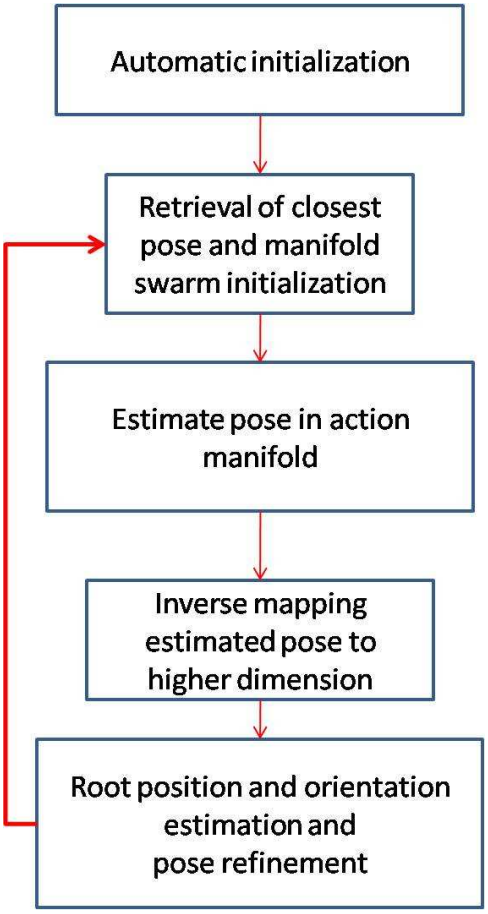



Figure 4.10: The tracking phase of our system

#### 4.3.2.1 Automatic Initialisation of 31D pose

We use the HPSO algorithm proposed in Chapter 3 to estimate the 31D pose in the first frame, owing to the automatic initialisation property of HPSO as demonstrated in Chapter 3. The initialised pose is then mapped to a 2D subspace co-ordinate on the pose subspace using charting's mapping function, and the particle swarm is initialised around it using a Gaussian distribution. Similar to the previous Chapter, the cost function for HPSO measures how well a pose hypothesis matches the multiview data from a set of synchronised cameras. In our system, for our studio sequence, we have two sets of multi-view data: silhouettes

and silhouettes without torso for accurate arm estimation.

#### 4.3.2.2 Subspace Pose Estimation

In order to estimate the pose in subspace, we propose a variation of PSO designed to polarise the search for the best next pose on or close to the subspace. The PSO framework was chosen for three reasons. First, it has been shown to estimate and track well 3D pose in high-dim joint angle space without motion models (but with long processing times, avoided in our system). Second, it is very easy to enforce nonlinear constraints on the search space. Third, a tested implementation was available to us. We first overview the steps involved in the standard PSO algorithm before presenting our modified PSO algorithm.

**PSO.**

1. **Initialisation**:

   - Initialise a population of particles $\{\mathbf{x}^i\}, i = 1 \dots N$, with positions randomly within search space $S$ and velocities randomly within $[-1, 1]$. For each particle evaluate the desired cost function $f$ and set $pbest^i = f(\mathbf{x}^i)$. Identify the best particle in the swarm and store its index as $g$ and its position as $\mathbf{p}^g$.

2. **Repeat** until the stopping criterion is fulfilled:

   - Move the swarm by updating the position of every particle $\mathbf{x}^i$, $i = 1 \dots N$, according to the following two equations:

$$\begin{aligned} \mathbf{v}_{t+1}^i &= \omega \mathbf{v}_t^i + \varphi_1(\mathbf{p}_t^i - \mathbf{x}_t^i) + \varphi_2(\mathbf{p}_t^g - \mathbf{x}_t^i) \\ \mathbf{x}_{t+1}^i &= \mathbf{x}_t^i + \mathbf{v}_{t+1}^i \end{aligned} \tag{4.16}$$

     where subscript $t$ denotes the time step (iteration).

- Ensure that $\mathbf{a} \leq \mathbf{x}^i \leq \mathbf{b}$. Search constraints are easily enforced through particle velocities. If the particle violates the search space boundary in some dimension, its position in that dimension is set to the boundary value and the corresponding velocity entry reversed.

- For $i = 1 \ldots N$ update $\mathbf{p}^i$, $pbest^i$, $\mathbf{p}^g$ and $gbest$.

**Subspace constrained PSO.** In subspace tracking, it is important to select a tracking framework, which ensures the search is performed near the learnt subspace structure, as, typically, the action is not well-defined or modelled at significant subspace deviations. The particle filtering framework is not suitable for subspace in this regard, as they have no principled mechanism to enforce search constraints, as we have described in the previous Chapter.

We address this issue by proposing a variation on the basic PSO, which we term the modified PSO. In our modified PSO, we replace the global best over the swarm, $p_t^g$, which changes at each iteration, with a fixed "target", the next point in the subspace in the direction of motion. This reduces the swarm mobility by forcing the particle to explore new poses in the direction of the evolution of the motion model, as desired. The final solution (pose estimate) is still chosen as the best particle in the swarm at convergence, which will be near the poses learnt but not necessarily one of them. In order to select the $p_t^g$ for a given frame, we use the refined 31D pose estimate in the previous frame, and retrieve its nearest neighbour index from the high-dimensional training dataset using a quick search. Given the retrieved high-dimensional nearest neighbour index, the corresponding next subspace index in time, functions as $p_t^g$.

We use a low inertia value to keep the search local. Finally, the search constraints $\mathbf{a}$ and $\mathbf{b}$, are set to the minimum and maximum spatial co-ordinates in each dimension. The search limits ensure the particles do not stray away significantly from the subspace. We believe this modification makes our proposed PSO tracker more suited for this problem than the particle filtering frameworks, which are

used in literature. We provide an illustration comparing the swarm behavior of the standard PSO algorithm, with the swarm behavior of the modified PSO algorithm in Figure 4.11.

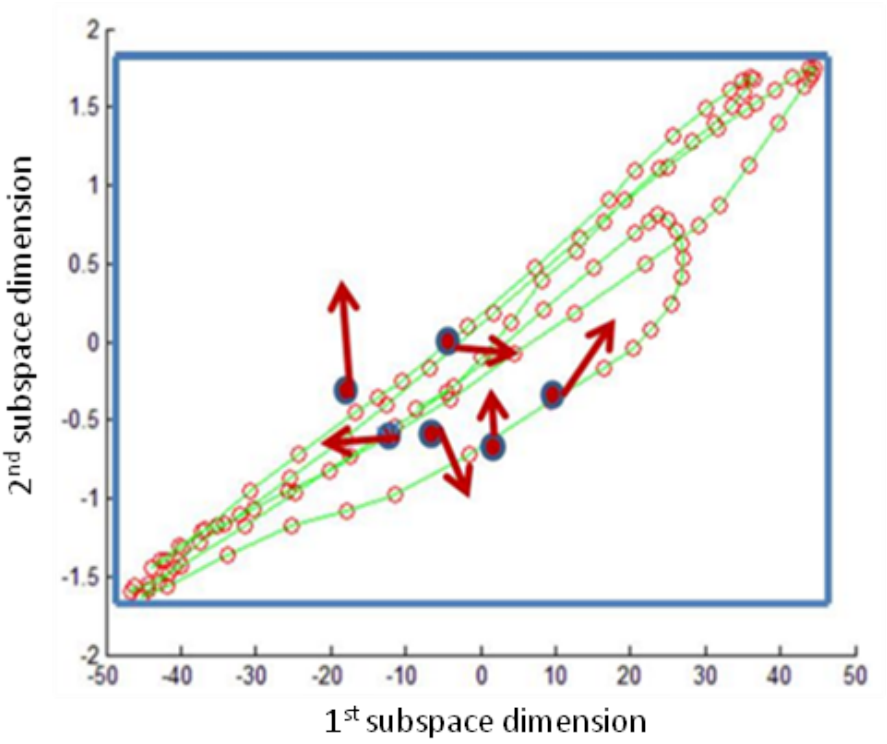
**Subspace Swarm Initialisation.** At each frame, a new PSO swarm is created whenever a new pose must be estimated. The previous best pose estimate in joint angle space is mapped to the subspace as initial pose (to initialise PSO search). To do this we perform a nearest-neighbour search in the high-dimensional joint angle space to retrieve the closest matching pose observed during learning, and map the pose found to the subspace. The swarm is then initialised using a Gaussian distribution and search performed.

**Subspace Pose Evaluation.** Our modified PSO is used to estimate the pose within the subspace. The subspace PSO hypothesis is evaluated using the learnt MVRVM mapping. The MVRVM mapping learnt during the training phase is used to transform the pose hypothesis from the action subspace into a set of shape context histogram, one for each camera. The similarity between the shape context histograms from observation and hypothesis is evaluated by Euclidean distance in each camera, and the overall, multiview similarity computed by summing over all the cameras.

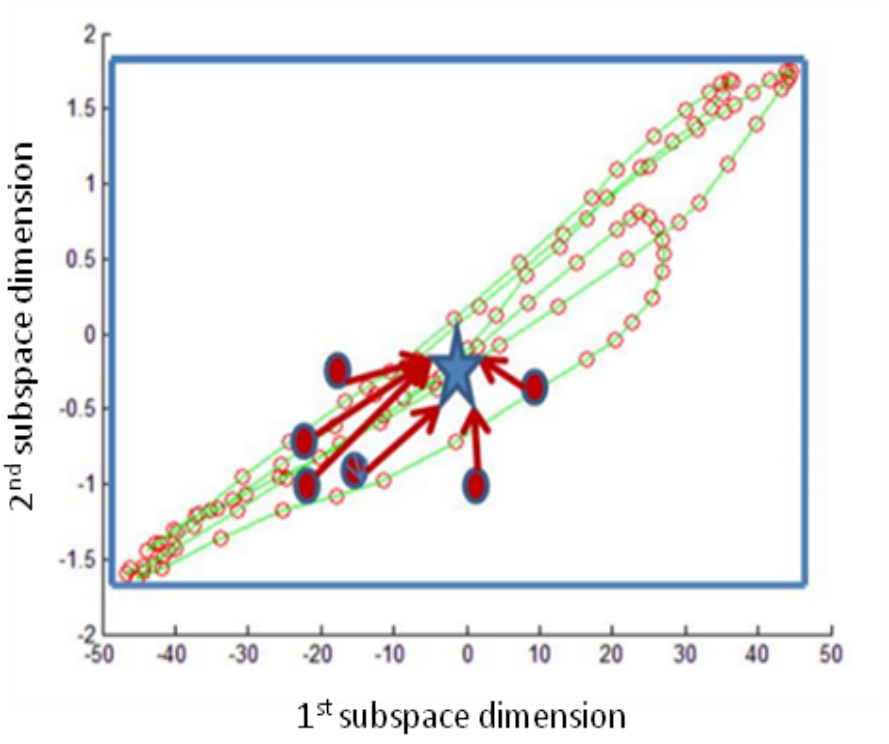
**Error Recovery.** The modified PSO, in addition to our instantaneous swarm initialisation, helps in avoiding divergence and aids in error recovery, as the swarm initialisation retrieves the closest pose in every frame, and subspace PSO constrains the search closer to the retrieved pose.

#### 4.3.2.3 Root Estimation and Pose Refinement

The best particle in the swarm at convergence gives the instantaneous best estimate of the pose in subspace. However the best estimate is a 2D subspace




(a)




(b)

Figure 4.11: Action subspace for the punch sequence, where blue bounding box denotes search limits and red circles denotes particles searching for optimum in a) standard PSO algorithm and b) subspace PSO algorithm, where blue star is the fixed global best particle, which constrains the search of the particles near the subspace.

coordinate, without the root position and orientation, which would be needed for applications like animation. This is addressed in our final step, where we obtain the corresponding estimate as 31D skeleton, by, firstly,mapping back from the subspace to joint angle space. Secondly, we need to add an estimate of root translation. Given the 31D skeleton, we adopt the HPSO framework to estimate the root position and orientation, and in addition to estimating the root position, we also perform a refinement of the final result, to accomodate subspace pose estimate errors. In our final HPSO framework, we exploit the fact that the back-mapped pose is close to the correct pose, and perform an efficient local search in 31D (including root translation and body orientation) with few particles, low inertia value and few PSO iterations. The result is the final pose estimate as joint angle vector, which is again mapped to the subspace to initialise the search in the next instant.

## 4.4 Experimental Results

In this section, we evaluate the performance of our proposed subspace tracking using three different experimental setups. Firstly, we compare our subspace tracking system with comparable state-of-the-art tracking algorithms, especially the Gaussian process annealed particle filter (GPAPF) [95], on our studio sequences and HumanEva dataset. In GPAPF, the subspace is learnt using GPLVM and tracking is performed using APF [95]. Secondly, we report a performance evaluation of our subspace system by varying the algorithm parameters. Finally, we compare the performance of the subspace system with HPSO using the HumanEva dataset.

*System Implementation.* Our proposed system is entirely based in MATLAB under Windows with 2.40 GHz processor. In our tracking framework, we imple-

mented the complete tracking phase and the charting component in the learning phase, while the SCH representation and MVRVM components where implemented using existing software.

### 4.4.1 Comparative Experimental Tests

#### 4.4.1.1 Datasets and Algorithm Parameters

We evaluated our subspace tracking system using two different datasets, our studio sequences and HumanEva dataset.

**Studio Dataset.** Our studio dataset was captured with 8 synchronised colour cameras with resolution 640 x 480 at 30 Hz. The camera was setup, so as to maximise useful silhouette information, while avoiding similar views. The joint angles were extracted from the multi-view silhouettes using HPSO. The extracted joint angles were then manually refined for every frame to be used as the training data for the learning phase. The manually refined joint angles are the training data for the learning phase and ground truths for the tracking phase. The training dataset consists of 300 frames for *walk* sequence, 200 frames for *punch* sequence, 250 frames for *body pose* sequence and 200 frames for *prayer* sequence. The test dataset consists of 2 subjects for each action sequence, except the *punch* sequence, which has 1 subject. The number of frames present in each test sequence are given in Tables 4.3 and 4.13. An example of our studio sequence is shown in Figure 4.12.

**HumanEva dataset**. The dataset contains multiple subjects performing a set of predefined actions with repetitions, using 7 cameras (3 colour and 4 gray scale cameras), and was originally partitioned into *train*, *validate*, and *test* sub-sets. We choose 4 actions: *walking*; *box*; *jog* and *gestures* performed by subjects S1, S2

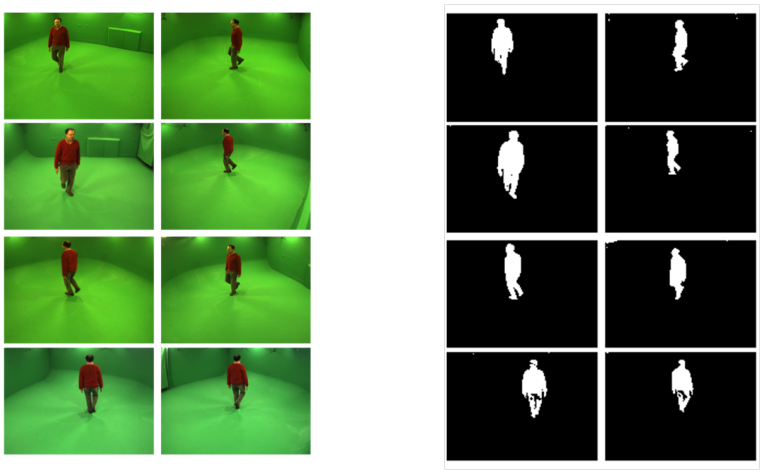


Figure 4.12: An example of our studio sequence

and S3. We trained our subspace tracking system using the *training* sub-sets of S2 and S3, containing approximately 1000 frames per action. Given the training dataset, we tested our subspace tracking system using the *validate* partition of S1, which is a subject not present in the training dataset. The *throw-catch* action is not selected as we have frequent frame drops, during the extraction of joint angles, using the tool provided by the HumanEva dataset itself. This action is not selected by other authors also [81].

**Learning Parameters.** Firstly, the sequences of multiple subjects were manually aligned to form a single sequence. The charting used 40 equally sampled charts from the dataset to reduce the 3D joint angle (without 6 dim root) to obtain the 2D joint angle subspace as shown in Figure 4.13. The intrinsic dimensionality for all actions was obtained as 2. We learnt separate mapping functions between the joint angle subspace and each 40D shape context histogram, for each camera using MVRVM run for 500 iterations with Gaussian kernel of width 0.5.

**Subspace Tracking Parameters.** The initialisation HPSO was run with 10 particles and the parameter settings described in Chapter 3. The PSO was run with only 5 particles for 30 iterations on the subspace space with a fixed inertia 0.5. The search limits for PSO are derived from minimum and maximum coordinates of the joint angle subspace. Finally the pose refinement was run with

5 particles for 20 iterations for each hierarchical step (12 hierarchical step). The starting inertia for refinement was 0.5, which amounted to a local search. The number of likelihood evaluations per frame amounted to 1350 (one evaluation per particle per iteration).

**GPAPF Parameters.** For our studio sequence, in order to ensure a fair comparison, GPAPF was run with 400 particles and 5 annealing layers to correspond to the number of likelihood evaluations (1350) of our proposed system. As the framework of GPAPF algorithm does not support automatic initialisation, we initialise the algorithm manually. We do not run the GPAPF on the HumanEva dataset, but report the results published in [97].

**3D Error Measure.** In our experiments, we use the error measure adopted in the previous chapter. The goodness of a pose estimate is obtained as a 3D error measure in millimeters, calculated as the average distance of 15 virtual markers, corresponding to Brown University software markers, on the pose estimate with respect to 15 virtual markers derived from the ground truth pose. In our studio sequence, the ground truth pose are obtained from manually refined HPSO pose estimates and motion capture-based ground truth poses in the HumanEva dataset. As the ground truth poses are manually refined from HPSO poses in our studio sequences, we do not use them to compare our subspace tracker with HPSO. Instead we use the HumanEva dataset to compare the two tracking algorithms.

#### 4.4.1.2 Results

**Accuracy.** In order to evaluate our proposed system and compare it with the GPAPF, the distance error between the manually extracted joint angles (ground truth) and pose estimates are calculated. The results in Table 4.2, Table 4.3 and

Figure 4.13, Figure 4.15 and Figure 4.14 suggest that at least in these experiments our proposed system is able to estimate the pose more accurately than GPAPF. Similar results are also obtained on the HumanEva dataset as shown in Table 4.7, where our proposed system performs better than GPAPF, APF, HPSO and tracking system proposed by Husz et al. [49].

**Computational Time.** Since we evaluate a pose hypothesis without expensive generation of silhouettes, our computational time is greatly reduced. The PSO search in subspace (initial estimate) takes 1 sec per frame and final refinement with root estimation takes 20 sec per frame. On the other hand, in the GPAPF algorithm, as evaluation of subspace hypothesis is performed by generating silhouettes from 3D cylinder, built by reverse mapping of subspace hypothesis to the high-dimensional space, the GPAPF algorithm takes 100 seconds per frame. The computational time taken by the different algorithms are shown in Table 4.7.

**Refinement of Poses.** Occasional wrong estimates are obtained from PSO in subspaces. This mainly occurs when the particles stray away from the subspace, inspite of the constraints. However as shown in Table 4.1, the refinement plays an important role in recovering from the wrong estimate, even when the particles stray away from the subspace, thus preventing the tracking from diverging (i.e., the inability to recover from wrong pose estimates and resume tracking correctly).

Table 4.1: Distance errors computed for subspace PSO estimate and final pose estimate, showing the effect of refinement step.

| Sequence (3 trials) | subspace PSO estimate | Final pose estimate |
|---|---|---|
| Vijay Walk | 29.7±20mm | 21.7±2.5mm |
| Jabez Walk | 28.7±7.6mm | 17.4±4.7mm |
| Adria Walk | 42.3±7.8mm | 34.29±7.7mm |
| Vijay Pose | 45.7±8.23mm | 28.8±4.28mm |
| Jabez Pose | 63.57±8.5mm | 40.58±13.63mm |
| Vijay Prayer | 33.91±9.29mm | 15.37±2.65mm |
| Jabez Prayer | 31.29±14.51mm | 17.54±9.74mm |
| Jabez Punch | 24.27±5.24mm | 12.40±3.12mm |

Table 4.2: Distance error computed for pose estimates of our system and GPAPF.

| Algorithm | Adria Walk | Jabez Walk | Vijay Walk |
|---|---|---|---|
| (3 trials) | (50 frames) | (50 frames) | (50 frames) |
| Charting PSO(accuracy) | 22.85 ±10.2mm | 17.4 ±4.77mm | 21.7±2.5mm |
| Charting PSO(time) | 25min | 25min | 25min |
| GPAPF(accuracy) | 35.14±13.34mm | 46.5±20.6mm | 34.4±13.8mm |
| GPAPF(time) | 1hrs40min | 1hrs40min | 1hrs40min |

Table 4.3: Distance error computed for pose estimates of our system and GPAPF.

| Algorithm | Charting PSO(accuracy + time) | GPAPF(accuracy + time) |
|---|---|---|
| Vijay Pose (55 frames) | 28.28 ±4.2mm (27min) | 40.2 ±6.62mm(1hr54min) |
| Jabez Pose (30 frames) | 40.58±13.3mm (15min) | 51.2±2.66mm(50min) |
| Vijay Prayer (30 frames) | 15.37 ±2.65mm(15min) | 26.76±7.29mm(50min) |
| Jabez Prayer(40 frames) | 17.5±9.74mm(20min) | 24.6±6.48mm(1hr12min) |
| Jabez Punch(30 frames) | 12.4±3.12mm(15min) | 21.19±1.74mm(50min) |

Figure 4.13: Tracking results of our system for Adria walk sequence displayed every 5th frame.

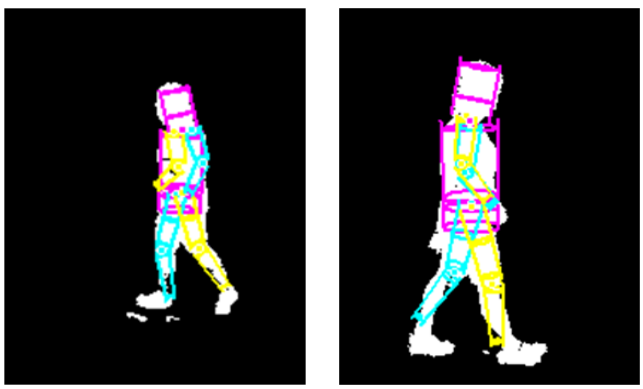


Figure 4.14: Tracking results of our system for HumanEva walk sequence, every 100th frame

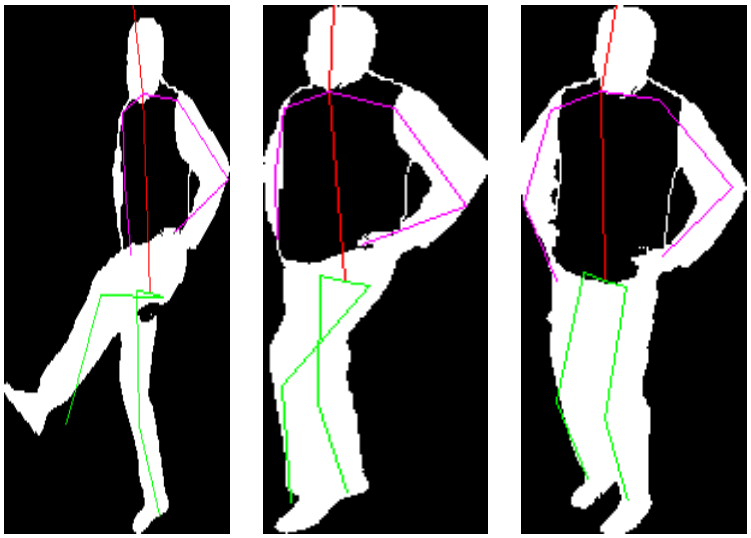


Figure 4.15: Tracking results of our system for Jabez kick sequence, every 10th frame

### 4.4.2 Performance Evaluation of Subspace Tracking

To evaluate the benefits of subspace evaluation and integration of subspace constrained PSO in our proposed algorithm, we ran a modified subspace tracking system with two important changes, as shown in Figure 4.16, firstly, we use the standard PSO algorithm for the subspace pose estimation. Secondly, we perform the subspace hypothesis evaluation by inverse mapping the hypothesis, building the 3D body model and generating expensive silhouettes. Henceforth, we refer to the modified subspace tracking system, as *PSO-Silhouette Evaluation Tracker* (PSO-S), and our original proposed subspace algorithm in Section 4.3.2, as *Modified PSO-Shape Context Descriptors Evaluation Tracker* (MPSO-SCH). The automatic initialisation of 3D pose and pose refinement in PSO-S, is done using the steps discussed in Section 4.3.2 (MPSO-SCH). The differences occur in the subspace pose estimation and evaluation step, which we next explain in greater detail.

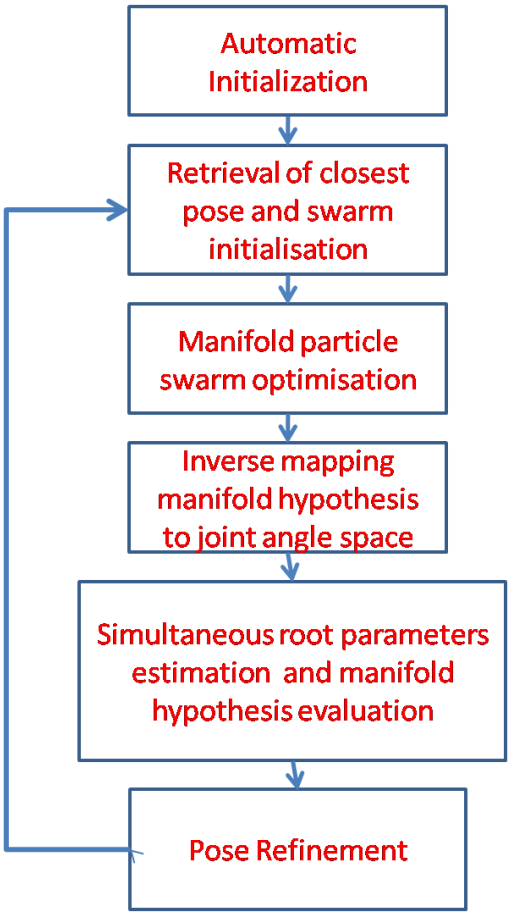



Figure 4.16: The tracking phase of our modified system

#### 4.4.2.1 PSO-S Subspace Pose Estimation

The subspace swarm initialisation is created, similar to MPSO-SCH, by performing a nearest-neighbour search and retrieving the closest matching pose. Given the initialised PSO swarm, we generate candidate hypothesis using the standard PSO, instead of subspace constrained PSO, as a result of which the particles are free to explore a wider search area, only constrained by the search limits.

To evaluate the candidate hypothesis, firstly we map the subspace particles back to the high-dimensional joint angle space (3D) using the inverse mapping learnt by charting. Secondly, rotation and translation parameters (6D) are added to the inverse mapped joint angles to obtain valid poses. 3D poses are used to generate

Table 4.4: Distance errors computed for subspace PSO estimate and final pose estimate, showing the effect of refinement step. The errors displayed correspond to the joint angles without the root co-ordinates.

| Sequence (3 trials) | Unrefined estimate | Final pose estimate |
|---|---|---|
| Vijay Walk (50 frames) | 26.97±4.10mm (50 sec) | 18.57±4.7mm(25min) |
| Jabez Walk (50 frames) | 24.78±4.87mm (50sec) | 15.4±4.05mm(25min) |
| Adria Walk (50 frames) | 39.5±7mm (50sec) | 31.77±5.87mm(25min) |
| Vijay Pose (55 frames) | 50.96±10.3mm (55sec) | 31.7±9.2mm(27min) |
| Jabez Pose (30 frames) | 59.3±11.1mm (30sec) | 39.67±11.63mm(15min) |
| Vijay Prayer (30 frames) | 28.7±3.31mm (30sec) | 25.60±3.3mm(15min) |
| Jabez Prayer (40 frames) | 45.18±14.22mm (40sec) | 24.54±5.55mm(20min) |
| Jabez Punch (30 frames) | 34.71±5.15mm (30sec) | 15.92±2.13mm(15min) |

a cylindrical 3D body model. Finally, the hypotheses are evaluated by selecting a predefined number of points on the surface of the 3D body model and projecting the surface points into multi-view silhouette observation. The goodness-of-fit is obtained in terms of the mean square error between the projected points and the silhouettes. The global best particle, $\mathbf{p}^g$, at the end of PSO iteration is considered to be the pose estimate for the given frame, which is then refined using HPSO with fewer iterations and particles. We report the performance of PSO-S below.

#### 4.4.2.2 Dataset and Algorithm Parameters

We evaluated PSO-S and MPSO-SCH using our studio sequences. We setup the algorithm using the tracking parameters defined in Section 4.4.1.1.

#### 4.4.2.3 Results

**Accuracy.** The results in Table 4.4 and Table 4.5 suggest that, at least in these experiments, MPSO-SCH is able to estimate the pose more accurately than PSO-

Table 4.5: Distance error computed for pose estimates of our modified PSO-based system and standard PSO-based system.

| Algorithm | MPSO-SCH | PSO-S |
|---|---|---|
| Adria Walk(50frames) | 22.85 ±10.2mm(25min) | 18.57±4.7mm(1hr40min) |
| Jabez Walk(50frames) | 17.4 ±4.77mm(25min) | 15.4±4.05mm(1hr40min) |
| Vijay Walk(50frames) | 21.7±2.5mm(25min) | 31.77±5.87mm(1hr40min) |
| Vijay Pose (55 frames) | 28.28 ±4.2mm(27min) | 31.7±9.2mm(1hr54min) |
| Jabez Pose (30 frames) | 40.58±13.3mm(15min) | 39.67±11.63mm(50min) |
| Vijay Prayer (30 frames) | 15.37 ±2.65mm(15min) | 25.60±3.3mm(50min) |
| Jabez Prayer(40 frames) | 17.5±9.74mm(20min) | 24.54±5.55mm(1hr11min) |
| Jabez Punch(30 frames) | 12.4±3.12mm(15min) | 15.92±2.13mm(50min) |

S in a majority of actions (*Vijay walk, Vijay pose, Vijay prayer, Jabez prayer, Jabez punch*), while reporting similar accuracy for the remaining actions (*Adria walk, Jabez walk and Jabez pose*). This can be attributed to the subspace search constraint, that we have introduced in our modified PSO. In the absence of such a constraint, the standard PSO is less likely to obtain similar results, a result of frequent deviation from the learnt subspace. It is worth noting that in spite of better performance, the modified PSO is formulated only for tracking with a prior motion model, with a fixed $\mathbf{p}^g$ at each instant. On the other hand, the standard PSO provides the solution for an optimisation problem, without any prior motion information, functioning as a generic optimisation algorithm.

**Computational Time**. Similar to the results observed in Section 4.4.1.1, our modified subspace system takes 100 seconds per frame, which is similar to GPAPF's computational time, owing to hypothesis evaluation using expensive silhouette generation. This demonstrates the computational advantage in evaluating our hypothesis in the subspace itself.

**Refinement of Poses.** The pose refinement step in our modified subspace tracking system does improve the tracking accuracy, similar to the behaviour observed in MPSO-SCH. This is shown in Table 4.4.

### 4.4.3 Experiments to Compare MPSO-SCH with HPSO

In this subsection, we evaluate the benefit of incorporating a motion prior, in the form of learnt sub-space, within the tracking framework. We perform a quantitative analysis using the HumanEva dataset, and report our observations based on the experimental results obtained.

#### 4.4.3.1 Dataset and Algorithm Parameters

We use the HumanEva-I dataset and perform our experiments on *walk, jog, gesture* and *box* actions. Our proposed subspace system was trained using subjects *S2* and *S3*, while testing was performed on subject *S1*. We used the HPSO parameter settings used in Chapter 3, and similarly, we used the algorithm parameters specified in Section 4.4.1.1.

#### 4.4.3.2 Results

**Accuracy.** In order to evaluate the subspace tracking system and compare it with HPSO tracking system, the distance error between the ground truth joint angles (ground truth) and pose estimates are calculated. The results in Table 4.6, and Table 4.7 suggest that at least in these experiments, MPSO-SCH performs marginally better than HPSO in a few sequences, while HPSO performs marginally better on a few sequences. Moreover, we also compare the performance of our tracking systems, both MPSO-SCH and HPSO system, with other state-of-the-art tracking systems on the *S1 walk* sequence. We obtain the error values, for *S1 walk* sequence, from the corresponding publications of

| Test Sequence (580frames) | HE-I (*S1 walk, validate)* |
|---|---|
| MPSO-SCH | 82mm(5hrs20min) |
| HPSO | 88mm(14hrs50min) |
| GPAPF [97] | 86.3mm(16hrs12min) |
| APF [97] | 95.4mm(19hrs33min) |
| Husz at al [49] | 101.8mm(NA) |

Table 4.6: Distance errors computed for HumanEva dataset.

| Test Sequence (HumanEva-I) | MPSO-SCH | HPSO |
|---|---|---|
| S1 Jog (517-667) | 80mm(1hrs25min) | 85mm(4hrs15min) |
| S1 Gestures (386-486) | 22mm(50min) | 20mm(2hrs50min) |
| S1 Box (396-496) | 70mm(50min) | 68mm(2hrs50min) |

Table 4.7: Distance errors computed for HumanEva dataset.

the tracking systems [97, 49], which do not report results on other sequences of *S1*. As shown in Table 4.6, our subspace tracking systems performs the best, being marginally better than the subspace tracking system (GPAPF). Though the performance of HPSO is weaker than the subspace systems, the tracking accuracy is better than APF as well as the tracking system proposed by Husz et al. ([49]).

**Computational Time.** Apart from the reduction in computational complexity owing to the subspace evaluation scheme, there exists an inherent reduction in computational time as a result of using motion priors, as effective constraints in the search space. In practical terms, this implies that the subspace based-PSO has learnt search limits, pertaining to each action. While in HPSO, as the search space is constrained by the generic biomechanical joint limits, rather than action-specific limits. This leads to an increased number of particles and search iterations in HPSO compared to subspace PSO.

### 4.4.4 Analysis of Experimental Results

*Comparison with State-of-the-Art.* Based on the experimental results, the tracking performance of our subspace framework (MPSO-SCH and PSO-S) is comparable and marginally better than comparable state-of-the-art tracking algorithms including GPAPF and APF. The tracking performance of our proposed algorithm can be attributed to the reduced search space, and the integration of HPSO, for initialisation and refinement. Additionally, the modified PSO plays an important role in the pose estimation, by constraining the search near the subspace, and incorporating subspace search limits, features not present in the original APF algorithm.

In addition to the comparable or marginally better tracking performance, an important feature of our subspace tracking algorithm is the greatly improved computational time, which varies between 1sec for subspace pose estimation and 20 sec for full-body estimation. Note that for applications, where the root position and orientations are either fixed or assumed, only the subspace pose estimation is required. The improvement in computational time can be attributed to our subspace evaluation scheme, in addition to the reduced subspace, resulting in fewer search iteration.

*Comparison of MPSO-SCH and PSO-S.* A comparative analysis between MPSO-SCH and PSO-S clearly demonstrates the benefits of selecting a modified PSO for subspace pose estimation, instead of the standard PSO, as the fixed target in each iteration serves to constrain the search near the expected subspace regions. Additionally, the computational advantage of our subspace evaluation scheme is demonstrated with the significant reduction in MPSO-SCH's computational time.

*Comparison of MPSO-SCH and HPSO.* Based on our experimental results, our proposed subspace tracking framework performance is comparable and marginally

better than HPSO, in addition to reduced computationl time.

*General.* Inspite of the advantages of subspaces tracking discussed above using a motion prior in the subspace tracking system has a few drawbacks when compared with systems like HPSO. Firstly, the subspace tracking system is an action-specific tracking system, i.e., prior knowledge of the action being tracked is required, whereas HPSO functions as a black-box system with fixed parameter settings for different actions. Secondly, HPSO can be used across different datasets, while the subspace tracking system can only be used in the studio environment from which the learning data is acquired, due to the camera invariant property of shape context descriptors-sensitivity to camera position in the studio. Summarising the two systems a trade-off between HPSO and MPSO_SCH exists, namely, the reduced computational time and increased accuracy of MPSO-SCH and the general black-box tracking property of HPSO at similar accuracy and increased computational time.

## 4.5 Conclusion and Future Work

In this chapter, we have presented a framework for markerless articulated human motion tracking in multiple-view sequences using charting to learn the low-dimensional subspace for common actions. To our best knowledge, charting is used within articulated body tracking for the first time. Additionally, tracking is performed in the subspace space using a variation of PSO, which enforces a soft constraint to keep the search close to the action subspace, still admitting reasonable departures from poses observed during learning. Our proposed tracking framework is able to track efficiently without learning explicit models of the subspace dynamics, avoid divergence and recover from wrong estimates. The use

of motion prior, greatly reduces the computational time when compared with HPSO-based tracking system. Moreover, we have demonstrated better performance than similar subspace tracking systems, GPAPF. However the drawback of incorporating the motion prior is the dependence on specific action and datasets. Addressing these issues would be the focus of our current and future work. The subspace tracking system can be made capture-environment invariant by using 3D shape descriptors such as [108], instead of 2D shape context histogram descriptors which depend on the camera view. A primary requirement of our current subspace tracking system is the manual selection of the subspace corresponding to the action being tracked; this can be avoided by incorporating an action recognition framework within the subspace tracking framework, which forms the basis of our next motion analysis system-subspace human motion classification system, explained in the next chapter.

# Chapter 5

# Classifying Multi-View Human Action Snippets using Charting and Action Subspace Features

## 5.1 Introduction

In the previous chapters (Chapter 3 and 4), we addressed an important area of research in human motion analysis, namely, human motion tracking, and extracting human motion information from video sequences. The extracted information is used in a wide number of applications including animation, sports analysis, and biomedical analysis. Another important area of human motion analysis is human motion classification, defined as the assignment of an action class label to video sequences. Some of the applications include video surveillance, object-level

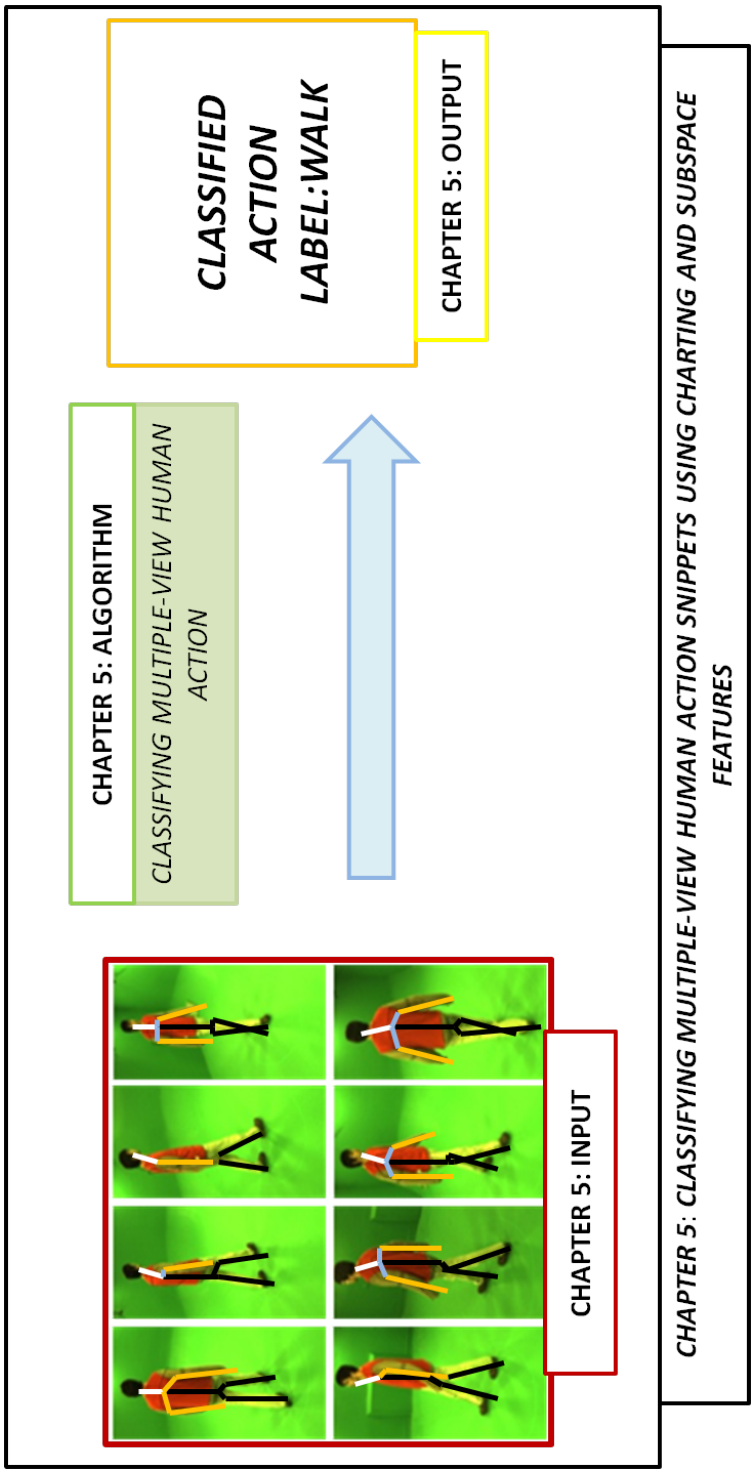



Figure 5.1: Overview of Chapter

video summarization and indexing.

In this chapter, we introduce a framework to classify full-body, markerless articulated human motion. The input are skeletal poses represented by a vector of joint angles obtained either from a commercial mocap system dataset [27] or from our automatic pose tracking system (Chapter 3 and 4). The pose estimated by a tracking algorithm in each frame corresponds to a coordinate in the subspace, and the entire estimated sequence of poses is represented by a curve in this subspace. The classification of the motion is based on the comparison of the sequences of subspace coordinates (subspace trajectories) to sequences that represent different actions. These subspace trajectories are used to classify human motion. Additionally, with on-line scenarios in mind, we identify the minimum sub-sequence length allowing reliable recognition (over the set of actions learnt in this work). We call such sub-sequences *snippets*.

Classifying articulated motion is an important problem and a challenging one because of the complex and generally unpredictable nature of human movements and the high-dimensional search space, with typically $20 - 50$ degrees of freedom for 3D skeletal pose ([47]) and $> 100$ degrees of freedom for image-based descriptors like shape context and HOG ([81, 25]). An important challenge posed by variations in motion style within the same action (intra-class variations); for example, a walk can vary in speed and style. Another challenge is the similarities between certain actions; for example, slow running is similar to jogging. A good human action classification approach should be able to generalize over intra-class variations, while providing good inter-class discrimination. In our work, we are particularly interested in addressing the issues of high-dimensional search, while simultaneously maintaining a high classification accuracy.

Typically, in a video-based human motion classification algorithm there are two main steps: extracting discriminative features from video sequence, and assigning an action label to extracted features from a set of predefined action class labels.

The most important step is the extraction of discriminative features, either *image-based features* [25] or *skeletal features* [66]. Image-based features are extracted directly from the video sequences, while the skeletal-based features are obtained either from motion capture data or as the output of a tracking algorithm. Several human motion classification algorithms are based on image-based features, where an important goal is the extraction and representation of discriminative features. Shape and motion information [64], 3D space time features [37], appearance and position context descriptors [81] are representative of the different discriminative features used in human motion classification algorithms.

In addition to the type of features, another paradigm used for categorising human motion classification is the size of the search space. Most papers in human action classification include high-dimensional feature representations, resulting in increased search complexity. However in recent years, the literature includes a number of motion classification systems learning low-dimensional, non-linear subspaces to address the issues of high dimensionality and complex human motion. Techniques for identifying low-dimensional subspaces in subspace include local linear embedding [52], GPVLM [59] and its variations, locality preserving projections (LPP) [141], and local spatio-temporal discriminant embedding [53].

Most such systems learn a single subspace from examples of multiple actions, [132, 109] so that adding new actions may decrease the degree of class discrimination, reduce consistency and smoothness of the subspace structure [96], make it difficult to establish useful embeddings with discriminative features, and increase embedding complexity [36]. These challenges have been the primary focus of attention in recent subspace classification systems, for example the discriminative GPLVM [132] and discriminative dimensionality reduction [109]. An alternate approach would be to create separate, action-specific subspaces, resulting in a straightforward approach to the above issues. This approach allows for an easier incremental addition of newer actions in subspace, whereas a single subspace

would have to be re-learnt. In this chapter, we propose, and evaluate, human motion classification systems using both single subspace and multiple subspaces.

*Context of Literature Classification.* In the context of our defined classification of human motion classification literature (Section 2.3.2), our final motion analysis work, presents markerless multiple-view human motion classification using charting to learn separate action-specific subspaces (**low-dimensional**) from 3D skeletal features (**features**). Moreover, we derive discriminative latent-space motion patterns and key frame-based representations, which are used in a multi-layered classification scheme.

*System Overview.* In our multi-view classification framework, the extracted skeletal features from the video sequences are represented using their subspaces. We adopt *charting* [15] to model the evolution of the angles of 3D skeleton in a low-dimensional sub-space. Charting, a dimensionality reduction technique not yet used in human motion classification, estimates automatically the dimensionality of the embedded subspace, preserves closeness of similar poses in the subspace. In order to assign an action label to the subspace features, we adopt a multi-layered classification framework in the subspace. The set of candidate actions (subspace) is pruned at each layer. In the final layer, where only similar classes with subtle variations remain, multi-dimensional dynamic time warping, a feature vector alignment algorithm [123], is used to classify the query snippet.

To the best of our knowledge, our work differs from the current literature in at least five ways. First, we investigate charting for action classification. Second, we derive a novel action representation based on subspace features. Third, we determine the minimum snippet length required for accurate classification for subspace skeletal features. Fourth, we use a compact representation based on key-frames in subspace for early pruning of action candidates. Finally, we compare and analyze subspace classification systems with single subspace and multiple subspaces.

We test our systems on HumanEva and CMU mocap datasets, achieving comparable or better classification accuracy than various comparable systems. Finally, we evaluate the single subspace and multiple subspace system and report our observations, based on the results obtained.

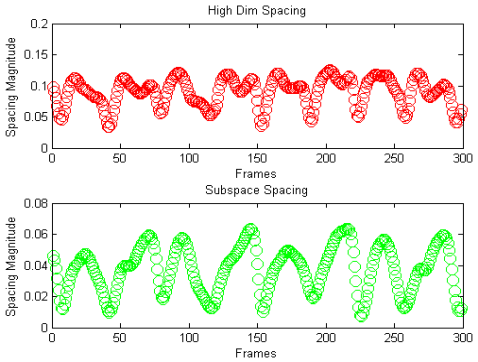
*Chapter Layout.* The rest of this paper is organised as follows. In Section 5.2, we explain about the first step involved in our multi-view classification framework, extraction of subspace discriminative features, obtained using charting. Next in Section 5.3, we present a brief overview of different distance metrics, which form the basis of our multi-layered classification system. Section 5.4 presents our single subspace classification framework, and our multiple subspace classification framework. Section 5.5 presents experimental results of our proposed system on HumanEva and CMU mocap Dataset. Section 5.6 summarizes our work and suggests future developments.

## 5.2 Charting-based Subspace Features

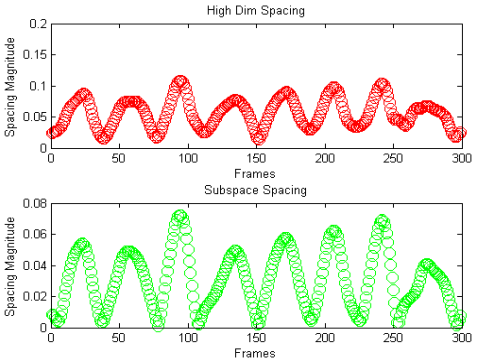
In Chapter 4, we provided a detailed overview of charting and used it within our subspace human motion tracking framework. Charting was used to model the evolution of 3D joint angles, obtaining a low-dimensional subspace representation. The learnt subspace representation was shown to preserve the geometry of high-dimensional local neighbourhoods in the subspace [15], a property which we used to propose a modified particle swarm optimisation to estimate the pose in the subspace. In this section, we provide an illustration of charting's inter-frame spacing preservation properties. Furthermore we also explain about the derived subspace features vectors used in our classification framework, before discussing about our key-frame-based subspace representation, used as a pruning layer in our multi-layered classification framework. Finally, we compare and analyse the single and multiple subspaces learnt from multiple actions.

*Inter-frame Spacing Preservation.* In addition to preserving the geometry of local neighbourhoods, charting also preserves the high-dimensional spacing between the poses, or frames, in the subspace. An illustration of the spacing preservation is shown in Figure 5.2. The spacing between high-dimensional poses are obtained as the distance between two position vectors, where each position vector corresponding to the vector defined between the origin and pose in a frame. Similarly, the spacing between subspace co-ordinates are also obtained as the distance between two position vectors. As shown in Figure 5.2, charting preserves the spacing between the poses in the subspace. The degree of spacing preservation is obviously increased, when the subspace is learnt for specific body parts. An example of which can be seen in Figure 5.2 (b-g). Figure 5.2 (b) shows the spacing preservation of subspace learnt for the upper body of the walk action and Figure 5.2 (c) shows the spacing preservation of lower body walk subspace. More examples of the same property is observed for the jog, box and gesture action. Charting's spacing-preservation property is used in our multi-layered classification framework, with dynamic time warping, to discriminate between similar actions, for example, walk and run. Specifically, they are used in our subspace feature vector, and as shown in our experimental section, the subspace feature vector in addition to the dynamic time warping increases the classification accuracy.

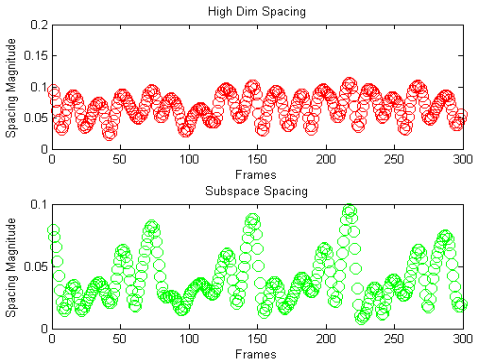
*Subspace Feature Vector.* We derive discriminative features in the action-specific subspaces, one for each action. We found by observation that discriminative high-dimensional 3D skeletal pose information is well captured, in 2D subspace, by two features: (a) absolute position of a point (pose in a frame), and (b) relative position of neighboring points (poses in successive frames), i.e., length of the vector connecting the two points. While the spatial co-ordinates alone are sufficient to classify very different actions like walking and punching, spacing greatly aid to distinguish similar actions like walking and jogging, especially in the multiple subspace framework. The training set of subspace feature vectors is



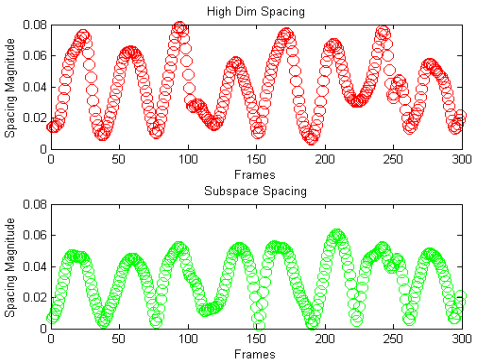

(a) Full Body subspace



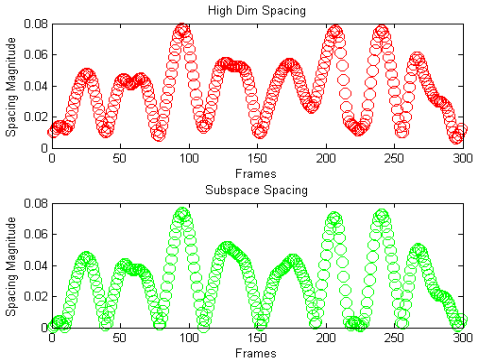

(b) Upper body subspace



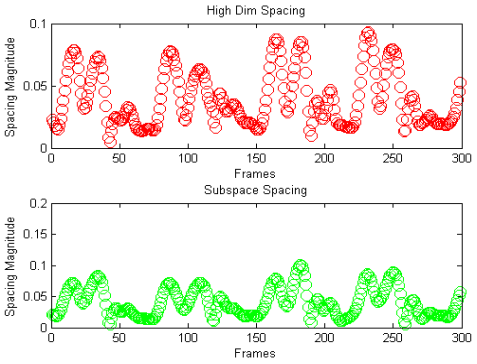

(c) Lower body subspace



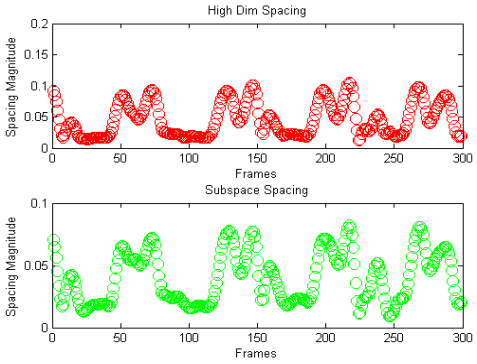

(d) Right arm subspace




(e) Left arm subspace

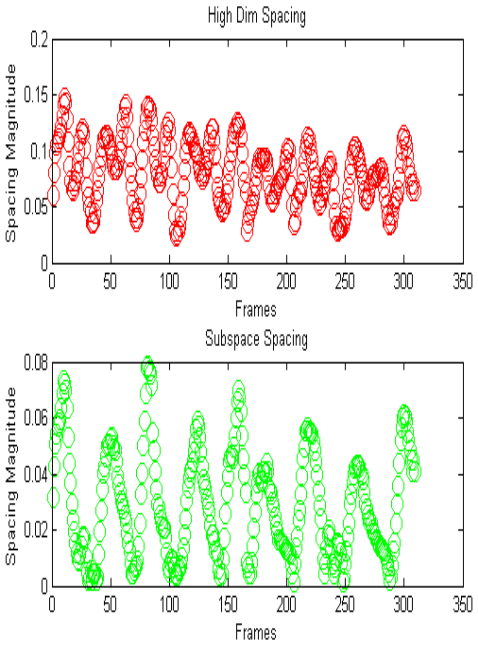



(f) Right leg subspace

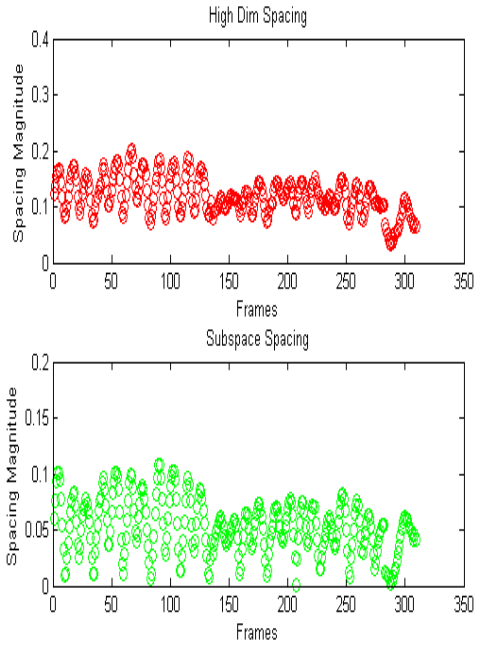



(g) Left leg subspace

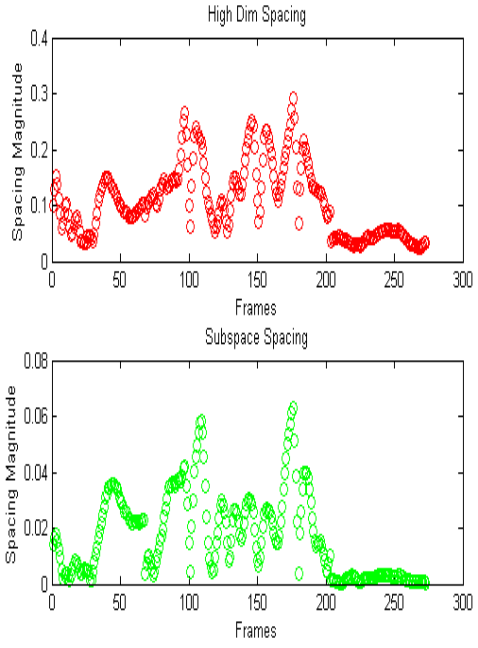
Figure 5.2: Illustration of spacing preservation in the learnt subspace of *Lee walk* action. Additionally the spacing preservation in hierarchical subspaces is shown (b-g), where spacing preservation is increased.



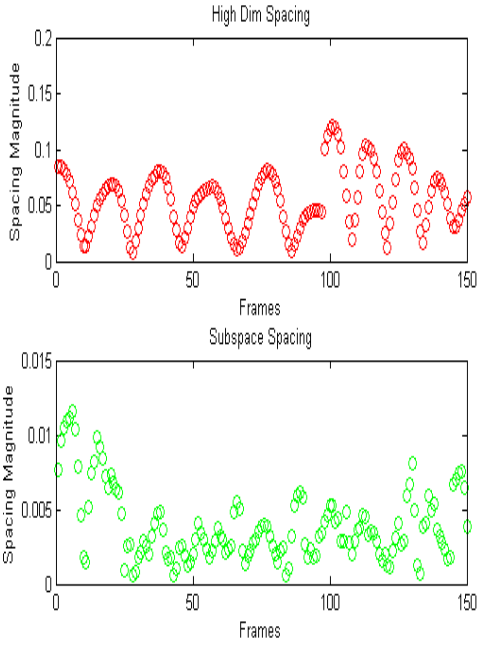

(a) Walk-Multiple subspace




(b) Jog-Multiple subspace

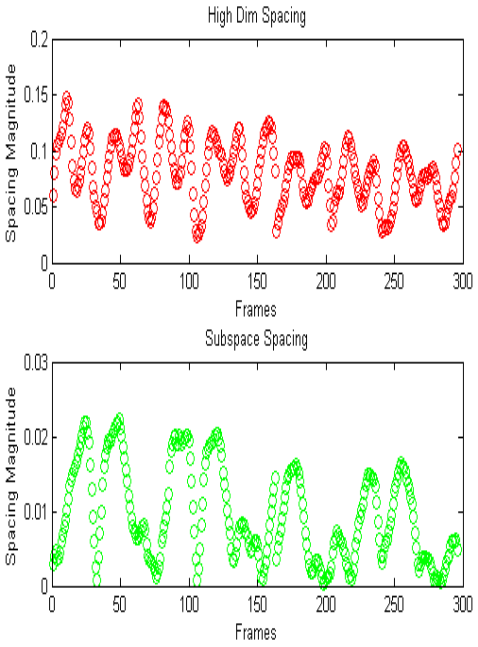



(c) Box-Multiple subspace

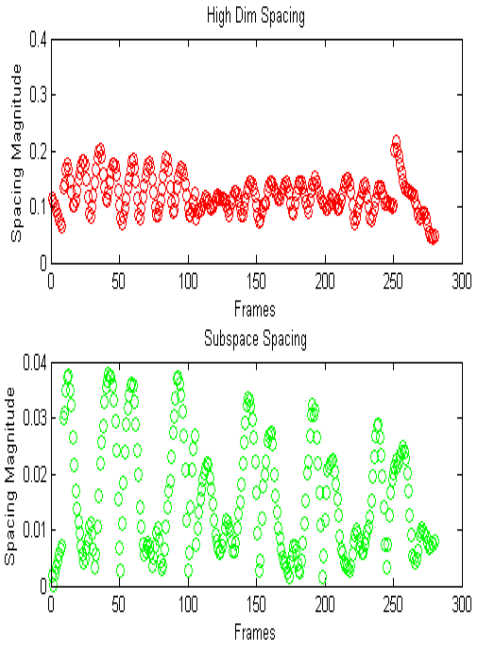



(d) Gesture-Multiple subspace

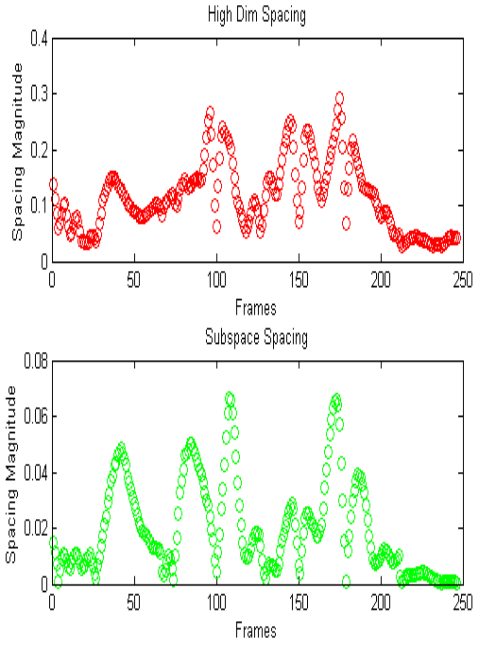
Figure 5.3: An illustration of spacing preservation for HumanEva action dataset on a multiple subspace (Walk, Jog, Gestures and Box).



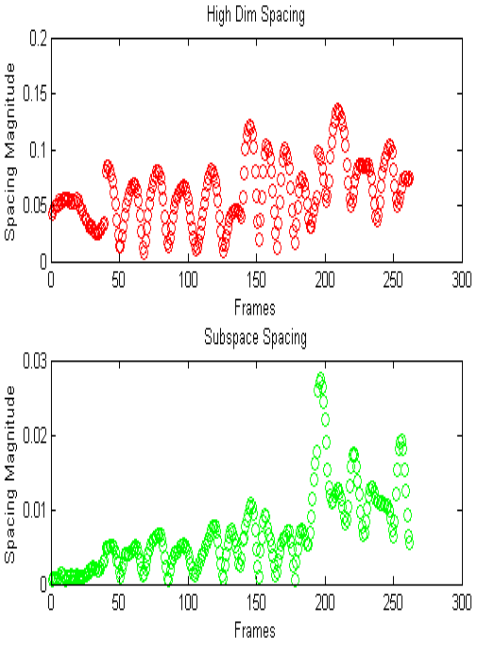

(e) Walk-Single subspace




(f) Jog-Single subspace




(g) Box-Single subspace




(h) Gestures-Single subspace

Figure 5.4: An illustration of spacing preservation for HumanEva action dataset on a single subspace(Walk, Jog, Gestures and Box).

given as, $\mathbf{V}_t^c = \{\mathbf{v}_i{}_{i=1}^{j}\}$ with position and spacing co-ordinates in the $c^{th}$ partition of the single subspace (single subspace classification) and $c^{th}$ separate subspace (multiple subspace classification). Each subspace feature vector $\mathbf{v}_t^c$ in the training dataset of a given partition, is represented as:

$$\mathbf{v}_t^c = [\mathbf{x}^c, \mathbf{s}^c] \tag{5.1}$$

where $\mathbf{x}^c$ gives point co-ordinates, and $\mathbf{s}^c$ is the spacing between neighboring points.

*Clustering of Feature Vectors.* To support layered classification, the subspace feature vectors training dataset $\mathbf{V}_t^c$ are clustered using K-means algorithms and the corresponding $j$ cluster centers are stored as $\mathbf{W}^c = \{\mathbf{w}_i{}_{i=1}^{j}\}$. The number of cluster centers are given in the experimental section.

Finally, each $\mathbf{v}_t^c$ subspace feature vector, in the training dataset $\mathbf{V}_t^c$, is assigned the label of its nearest cluster center $j$. The clustering of feature vectors is performed to reduce the computational complexity, by storing fewer examples from the training dataset. Additionally, the assignement of labels to each subspace feature vector is necessary to identify vectors from similar clusters for a detailed comparison in the final layers, as explained in detail in layer 3 of Section 5.4.2 and layer 2 of Section 5.4.1,

*Key-Frames Representation.* To obtain a concise representation of actions, we compute a set, $\mathbf{L}^c$, of key-frames in subspace, for each partition. As in animation, we define our key-frames as the starting and ending points of low-curvature subspace regions in subspace, currently using an empirical threshold on the angles between consecutive vectors. Such regions of high curvature (key-frames) correspond to instants in which an action changes significantly (see example in Figure 5.5). Finally we generate a feature vector,$\mathbf{v}_f{}^c$, of the key frames identified, in the same way as in Eqn (5.1).

$$\mathbf{v}_f{}^c = [\mathbf{x}_f{}^c, \mathbf{s}_f{}^c] \tag{5.2}$$

The set of key frame-based feature vector, $\mathbf{L}^c$, gives the concise representation of the $c^{th}$ space and is used in our layered classification scheme to prune the candidate actions.

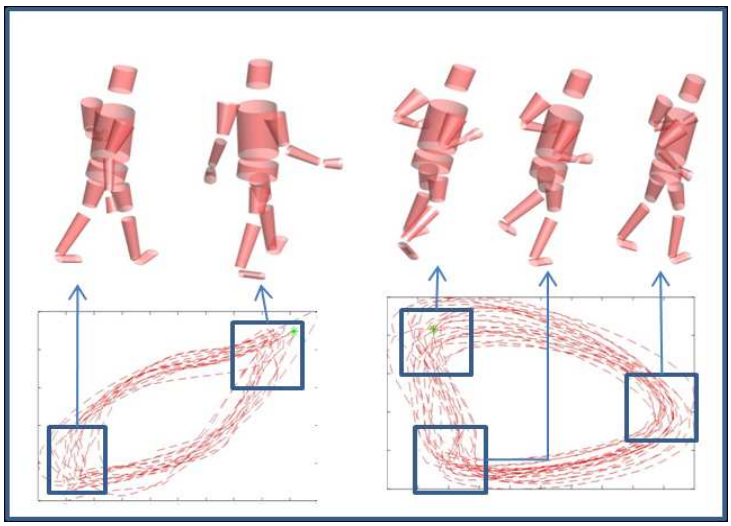
*Multiple and Single Subspace.* An important consideration for subspace-based human motion classification algorithms is the number of subspaces. Recent subspace-based human motion classification algorithms tend to learn a single subspace for multiple actions, where the primary focus of attention is to increase the discrimination among the classes. In our charting-based multiple and single subspace, as illustrated in Figure 5.6, we observe that the subspace structure demonstrates increased smoothness and consistency in the multiple action subspace, as a result of which, the extraction of key-frames representations are easier in this approach.

## 5.3 Distance Measures

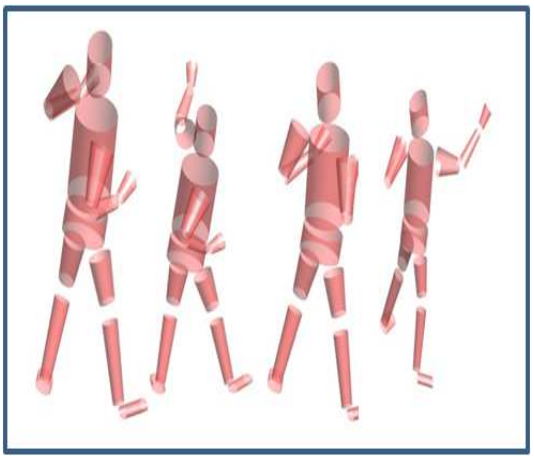
In this section, we provide a brief overview of Euclidean and Hausdorff distance metrics. Additionally, we explain about dynamic time warping and its extension, the multi-dimensional dynamic time warping.

*Euclidean Distance.* The Euclidean distance is the simplest and most widely used distance metric in various machine learning and computer vision applications. In Cartesian coordinates, if $\mathbf{p}$ and $\mathbf{q}$ are two vectors in $d$-dimensional space, then $d(\mathbf{p},\mathbf{q})$ is given by:

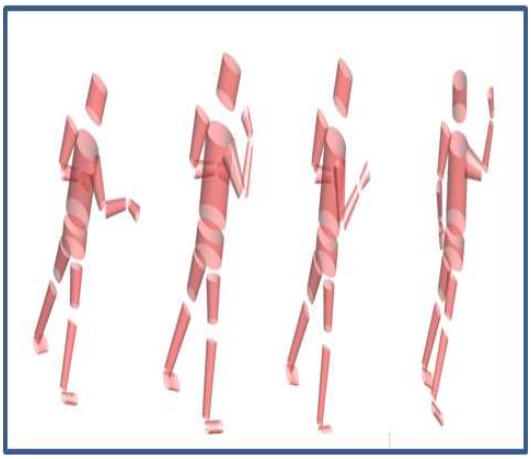
$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(q_1 - p_1)^2 + \ldots + (q_d - p_d)^2} \tag{5.3}$$



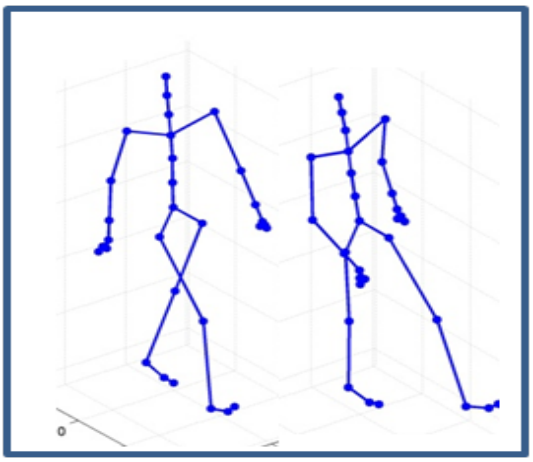
(a) Walk and Jog-HumanEva



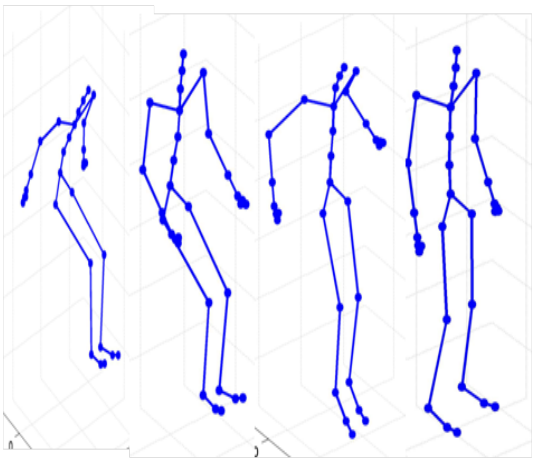
(c) Box-HumanEva



(d) Gestures-HumanEva

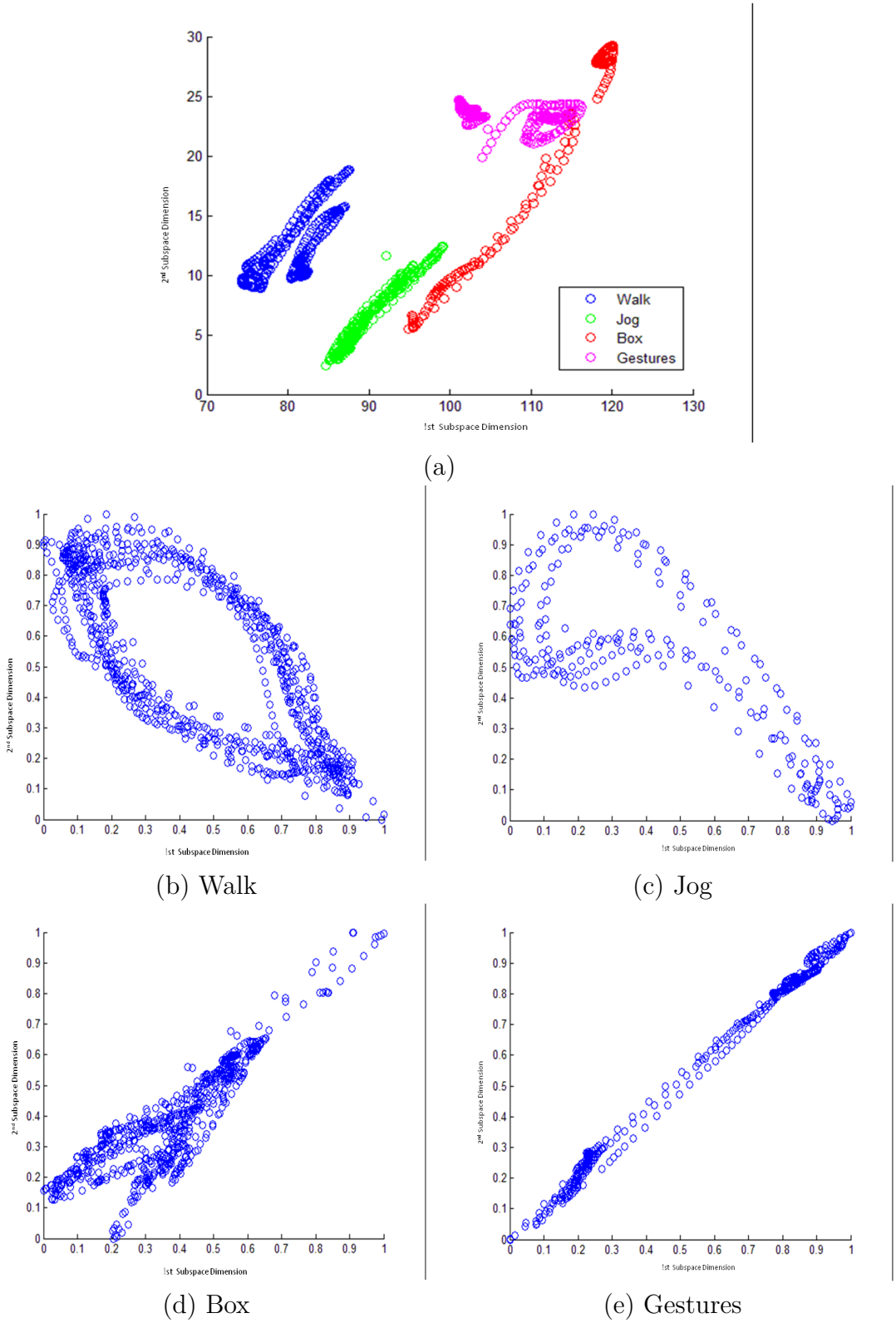


(e) Walk-CMU



(f) Jump-CMU

Figure 5.5: Examples of key-frames extracted from multiple subspaces (Separate subspaces learnt for each action)




(a)

(b) Walk

(c) Jog

(d) Box

(e) Gestures

Figure 5.6: A comparison of (a) single subspace and (b-e) multiple subspace learnt for HumanEva

*Hausdorff Distance.* The Hausdorff distance is a metric between two point sets. Given two point sets $\mathbf{A} = \{\mathbf{a}_k\}_{k=1}^{n}, \mathbf{a}_k \epsilon \mathbb{R}^{\mathrm{d}}$ and $\mathbf{B} = \{\mathbf{b}_k\}_{k=1}^{n}, \mathbf{b}_k \epsilon \mathbb{R}^{\mathrm{d}}$ the Hausdorff distance between them is defined as:

$$H(\mathbf{A}, \mathbf{B}) = max(h(\mathbf{A}, \mathbf{B}), h(\mathbf{B}, \mathbf{A})) \tag{5.4}$$

$$h(\mathbf{A}, \mathbf{B}) = max_{\mathbf{a}\epsilon\mathbf{A}} min_{\mathbf{b}\epsilon\mathbf{B}} \|\mathbf{a} - \mathbf{b}\| \tag{5.5}$$

where *h(***A, B***)* is called the directed Hausdorff distance from set **A** to **B**, which we use in our multi-layered framework. While the Euclidean distance metric measurement is a distance metric between any two point, the Hausdorff distance is a distance metric between any two point-sets.

*Multi-Dimensional Dynamic Time Warping.* Dynamic time warping (DTW) is a distance metric, between two sequences of possible varying lengths, which aligns the two sequences and obtains the ideal warp, or synchronisation, which minimises the distance between them. The optimal alignment is calculated using dynamic programming. A brief overview of DTW and its extension, the multi-dimensional dynamic time warping is given.

Given two sequences $\mathbf{A} = \{\mathbf{a}_k\}_{k=1}^{n}, \mathbf{a}_k \epsilon \mathbb{R}^{\mathrm{d}}$ and $\mathbf{B} = \{\mathbf{b}_k\}_{k=1}^{m}, \mathbf{b}_k \epsilon \mathbb{R}^{\mathrm{d}}$, where $d$=1 and length is $m$ and $n$. Firstly, DTW algorithm creates a $m$-by-$n$ local distance matrix **C,** where each ($i^{th}$ ,$j^{th}$) element represents the distance d($\mathbf{a}_i$,$\mathbf{b}_j$). Next, an accumulated distance matrix **D** is created, to accumulate the total distance between each possible pair of points of the two 1-dimensional sequences. The total distance in each matrix element ($i^{th}$ ,$j^{th}$) in **D** is obtained from the sum of ($i^{th}$ ,$j^{th}$) value in **C** and the smallest neighbouring accumulated distance. The main objective of DTW is use to obtain a warping function that minimizes the total distance between respective points of the sequence. This is is achieved by

finding the path with least cost, or smallest accumulated distances in **D.** The shortest possible path starts at the right bottom of the matrix D and goes to the left-top element in it, representing the best synchronization between the two sequences.

The original DTW was designed for one-dimensional sequences only, but there are many applications, as in our case, where the sequences are multi-dimensional. Recently, Holt et al. [123], proposed an extension of an extension of the original dynamic time warping algorithm (DTW) [123], known as multi-dimensional dynamic time warping (MD-DTW). In MD-DTW, the distance matrix **C** is obtained by computing the distances between $k$-dimensional points of the two sequences, unlike the original DTW algorithm, where **C** is calculated from 1-dimensional sequences. Furthermore, in MDDTW the input sequences are pre-processed before computing **C**. Specifically, each dimension in **A** and **B** is separately normalised to zero mean and unit variance. The normalisation step is necessary to obtain a distance measure, where each dimension is comparable. Finally, similar to the original DTW algorithm, given **C**, the accumulated matrix **D** is calculated, from which the path with least cost or smallest accumulated distance is obtained (*md-dtw distance*).

## 5.4 Subspace Classification Framework

In order to classify a test human action sequence, which forms a trajectory in the subspace, typically, two categories of algorithms exist. In the first group of algorithms, the focus is on enhancing the discrimination among the classes during the subspace learning process. Discriminative GPLVM and sufficient discriminative dimensionality reduction algorithms belong to this category. In the second group of algorithms the inter-class distance is not enhanced, instead computationally expensive trajectory, or curve matching algorithm, like the Frechet distance [96] or dynamic time warping [123] is incorporated within the classification frame-

work. Our work belongs to the latter category, where we use multi-dimensional dynamic time warping, in a multi-layered classification framework, to match the embedded test trajectory with the learnt subspace trajectories. Although we adopt charting to learn the subspace representation of actions, which does give a degree of discrimination among the classes, we do not focus on enhancing the inter-class distance. In our work, the subspace of actions is learnt in two different ways, firstly, a single subspace is learnt for multiple actions. Secondly, multiple subspaces are learnt for multiple actions (single subspace for each action). As illustrated in Section 5.2, few variations occur in the single and multiple subspace structures, which is accounted for in our multi-layered classification framework.

*Classification Overview.* In our classification framework, we propose a multi-layered classification framework, where the main motivation is to perform a layered pruning of candidate actions. Specifically, in our initial layers we compare our test human action sequence with the entire training dataset, containing candidate actions, using a computationally inexpensive search, and prune candidate actions which are dissimilar. Given the set of pruned candidate actions, we employ a detailed search using dynamic time warping. The multi-layered classification can be summarised as a scheme, where the search complexity is increased with a simultaneous reduction or pruning of candidate actions, thus functioning as a computationally efficient scheme.

Additionally we setup our mult-layered classification scheme in both the single subspace as well as the multiple subspace, primarliy to evaluate the two types of subspaces and analyse the corresponding classification results. We next explain about our single subspace multi-layered classification framework, before providing a detailed overview of the multiple subspace multi-layered classification framework.

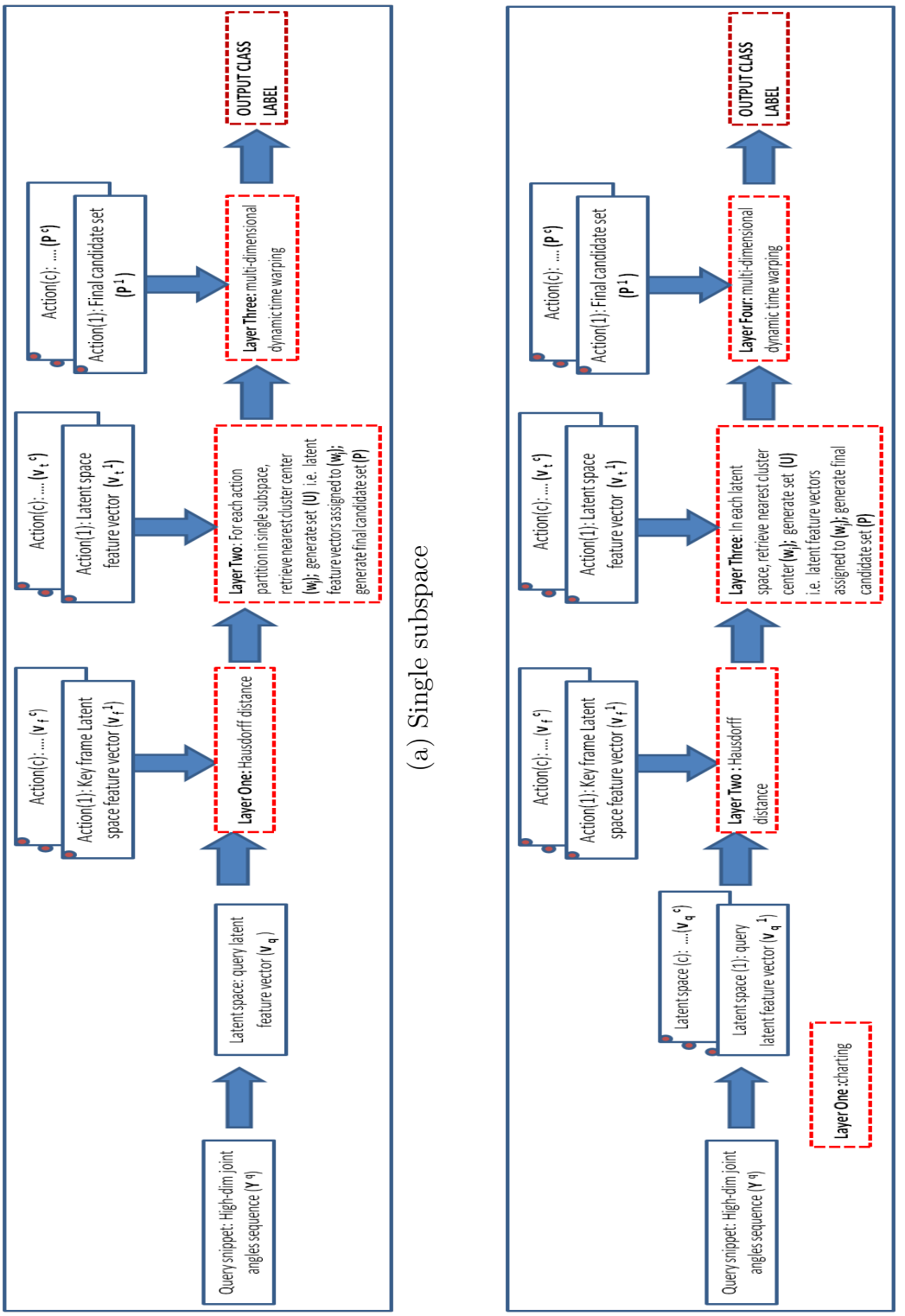
### 5.4.1 Single Subspace Multi-layered Classification

In our single subspace, we adopt a three-layered classification scheme with candidate action labels $v$ in the single learnt subspace $v = [v_1, v_2, ..., v_C]$, where $C$ is the number of action classes in the single subspace, which are pruned successively till the the correct action class $v_o$, corresponding to output class label is obtained. The multi-layered scheme provides an efficient search; the more similar two modelled actions, the more effort is needed to assign the query to one of them. So classification may terminate successfully after any layer, depending on the query and the dictionary of actions learnt. While our search, after test snippet mapping, is increasingly detailed at each layer, the number of search-candidate features vectors are simultaneously reduced, resulting in near-uniform computational time across layers, shown in Section 5.5.

*Overview of Framework.* Our classification scheme has three layers, after the mapping of the query snippet as shown in Figure 5.7 (a). The first layer performs a simple, quick search using key-frames, compared by the point-to-set Hausdorff distance. The second layer allocates the query to the nearest cluster in each independent subspace. The final layer uses multi-dimensional dynamic time warping to obtain the final output class. We describe the different layers in detail below.

#### 5.4.1.1 Classification Layers

*First Layer.* The classification process begins with mapping a query snippet $\mathbf{Y}^q = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n]$, with $\mathbf{y}_k \in R^D$ and $n$ number of frames, to the single subspace and generating $\mathbf{v}_q$, the query subspace feature vector. Once mapped, we exploit key-frames to prune candidate actions with a simple, quick search. We compute the point-to-set Hausdorff distance, $\eta^c$, between $\mathbf{v}_q$, and the set of feature vectors of key-frames for each action $c$, $\mathbf{L}^c$. Intuitively, in our case, this finds the distance of the nearest key-frame of each action's subspace to the query. The effect is to




(a) Single subspace

(b) Multiple subspace

Figure 5.7: An overview of our classification framework

eliminate actions for which the mapped query lies significantly away from the action subspace, without employing a detailed search. The rule is

$$\eta^{least} = \min_{c} \eta^{c}, \tag{5.6}$$

$$\upsilon_c = \begin{cases} 0 & if(\eta^c - \eta^{least}) > \varepsilon, class\ pruned \\ 1 & class\ not\ pruned \end{cases} \tag{5.7}$$

where $\varepsilon$ is set to $2 * \eta^{least}$ for all our experiments.

*Second Layer.* For each remaining candidate action, we compute the Euclidean distance between the query latent feature vector $\mathbf{v}_q$ and set of cluster centers $\mathbf{W}^c$ and retrieve the nearest cluster center $\mathbf{w}_j$. We then retrieve all subspace feature vectors $\mathbf{v}_t^c$ belonging to $j^{th}$ cluster, with cluster center $\mathbf{w}_j$ and obtain a set of $u$ subspace feature vectors $\mathbf{U}^c = \{\mathbf{v}_{i i=1}^{u}\}$. The cluster label associated with each training subspace feature vector is used for this operation.

Next we compute the similarity of $\mathbf{v}_q$ with $\mathbf{U}^c$ using the Euclidean distance and retrieve the $p$ nearest subspace feature vectors in $\mathbf{U}^c$ and form the set $\mathbf{P}^c = \{\mathbf{v}_{i i=1}^{p}\}$. This set of candidates is passed to final layer, for the final detailed search.

*Third Layer.* In this layer we use MDDTW, which finds the optimal alignment between $\mathbf{v}_q$ and latent feature vectors in $\mathbf{P}^c$. The final output class label ($\upsilon_o$) is then obtained as the action class $c$, whose candidate set $\mathbf{P}^c$ gives the smallest md-dtw distance. Though MDDTW improves the classification accuracy (Section 5.5), the computational cost is very high and not suited for an exhaustive search on all actions.

### 5.4.2 Multiple Subspace Classification

The main challenges of single subspace-based classification systems, with increase in number of actions, include decrease in degree of class discrimination, reduction of consistency and smoothness of the subspace structure [96], difficulty in establishing useful embeddings with discriminative features, and increased embedding complexity. Moreover, the subspace needs to be re-learnt for a new action. An alternate approach would be to create separate, action-specific subspaces, addressing the above issues. We next provide an overview of our multi-layered multiple subspace classification framework.

*Overview of Framework.* In our multi-subspace classification framework, we adopt the classification scheme used for the single subspace framework, with two extensions, effectively functioning as a four layered classification system. Firstly, to ensure a fair comparison across the different, independent subspaces, we scale-normalise the subspaces before deriving the subspace feature vectors. Our classification scheme has four layers, as shown in Figure 5.7 (b). The first layer performs mapping of the query snippet (high-dim joint angle subsequence) using charting, generating a query feature vector in subspace. The second layer performs the key-frames-based point-to-set Hausdorff distance. The third layer identifies the nearest cluster in each independent subspace. The fourth layer uses multi-dimensional dynamic time warping to obtain the final output class. We next explain about the first layer in detail, before summarising about the remaining layers, which are similar to the layers in the single subspace framework.

### 5.4.3 Classification Layers

*First layer.* Processing begins with mapping a query snippet $\mathbf{Y}^q = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n]$, with $\mathbf{y}_k \in R^D$ and $n$ number of frames, to the subspace of each candidate action class using charting. Snippet mapping performs an evaluation of the snippet

within the charts of every candidate action via Eqn (4.9). Query snippets evaluated with charts belonging to similar action classes will have a higher sum of probability, while charts belonging to dissimilar action classes would have near zero or zero sum of probability, as they would be treated as outliers. Action classes with low probabilities are pruned from the set of candidate actions and the query snippet is not mapped to the particular subspace. The query subspace feature vector ${\mathbf{v}_q}^c$ is then generated from the mapped query snippets in the remaining candidate action classes.

When a query snippet is mapped, we estimate the probability $e_k^c = p(k|\mathbf{Y}^q)^c$, where $\mathbf{Y}^q$ is the query snippet, that the data point belongs to each $k^{th}$ chart in action $c$. Using the parameters learnt (i.e., covariance $\Sigma_k$ in Eqn (4.9)) directly could result in poses with significant style changes not being recognised; for example, an exaggerated right hand forward pose of the *walk* cycle may not be recognised as *walk*, if it was not present in the training data. We address this problem by inflating the entries of the covariance matrix by 10% (uniformly increasing scale), which performed well in our experiments. The classification rule is

$$\upsilon_c = \begin{cases} 1 & if \ \sum_k(e_k^c) \gg 0 \\ 0 & if \ \sum_k(e_k^c) \simeq 0 \end{cases} \tag{5.8}$$

where $\Sigma_k$ refers to the sum over $k$ charts and not covariance.

*Second-fourth layers* are similar to the first three layers of the single subspace classification framework. However, unlike the single subspace framework, we have $c$-separate query subspace feature vectors ${\mathbf{v}_q}^c$ . Summarising the similarities in two systems. Firstly, the mapped query snippet is compared with the key-frame-based subspace feature vector in each subspace, and the point-to-set Hausdorff distance is used to prune dissimilar subspaces. Secondly, in the re-

maining subspaces, for the mapped query snippet, the nearest cluster center and its associated subspace feature vectors are retrieved, from which a few candidate subspace few vectors are used to form the final candidate set. Finally, we compute the mddtw distance between mapped query snippet and final candidate set in the remaining subspaces to finally obtain the correct action label.

## 5.5 Experimental Results

In this section we evaluate the performance of our proposed subspace classification framework by comparing the classification accuracies of the multiple subspace classification framework with the single subspace classification framework on the CMU mocap dataset [27] and HumanEva dataset [114]. Additionally, we compare the classification accuracies of our system with comparable state-of-the-art classification algorithms. We then report a performance evaluation of the multiple subspace and single subspace by varying the algorithm parameters. Our proposed system is entirely implemented in MATLAB under windows with 2.40 GHz processor, without using any existing softwares.

### 5.5.1 Comparative Experimental Results

#### 5.5.1.1 HumanEva Experiments (Dataset and Algorithm Parameters)

**Dataset.** HumanEva dataset contains multiple subjects performing a set of predefined actions with repetitions, and was originally partitioned into $train, validate$, and $test$ sub-sets. We choose 4 actions: $walking, box, jog$ and $gestures$ performed by subjects $S1, S2$ and $S3$. We use the $train$ partition of the S1, S2 dataset and test our sequences using $validate$ partition of all three subjects. Our test bed choices are similar to the reported tests in Ning et al.

[81] and Zsolt [47]. The *throw- catch* action is not selected as we have frequent frame drops, during the extraction of joint angles, using the tool provided by the HumanEva dataset itself. This action is not selected by other authors also [81].

**Training Parameters.** We used 20 charts to reduce the 26D skeletal angles (without 6D root) to separate 2D joint angle subspaces (multiple subspace) and single 2D joint angle subspaces (single subspace). The intrinsic dimensionality selected by charting was 2 for all actions. To obtain a smooth subspace, we remove subsequences of the training data set with joint angle discontinuities which could be attributed to bad or inconsistent marker data. Finally we smooth the pruned data set. The result is $2000 - 3000$ frames per action in the training dataset.

To determine the minimum length of action snippets required for consistent and reliable classification accuracy, we varied the length of $\mathbf{x}$ in $\mathbf{v}$ within the set $[3, 8, 16, 30, 50, 75]$. $\mathbf{s}$ is subsequently derived for each $\mathbf{x}$ length, to form the concatenated feature vector $\mathbf{v}$. This is also done for $\mathbf{v}_f$ in Eqn (5.2). Furthermore, we obtain 10 clusters, $\mathbf{w}$, in set $W$ (*layer* 3, Section 5.4.2 and *layer* 2, Section 5.4.1) and the number of candidate latent feature vectors in $P$ (*layer* 3, Section 5.4.2 and *layer* 2, Section 5.4.1) is 3. Finally in each subspace, we derive $20 - 40$ key-frames, located at the regions of high-curvature, a sample key-frame from each high-curvature region is shown in Figure 5.5 for all actions.

#### 5.5.1.2 HumanEva Experiments (Results)

**Classification Accuracy.** We evaluate both our classification systems with $500 - 1500$ test action snippets per action and owing to the difference in the nature of comparative algorithms, we compare with the best accuracies reported by each system, treating the different classification algorithms as a black box.

*Ning et al.* [81]. The average classification accuracy for query snippets of varying length is shown in Figure 5.8 and Table 5.1, where our result with 75-frame action

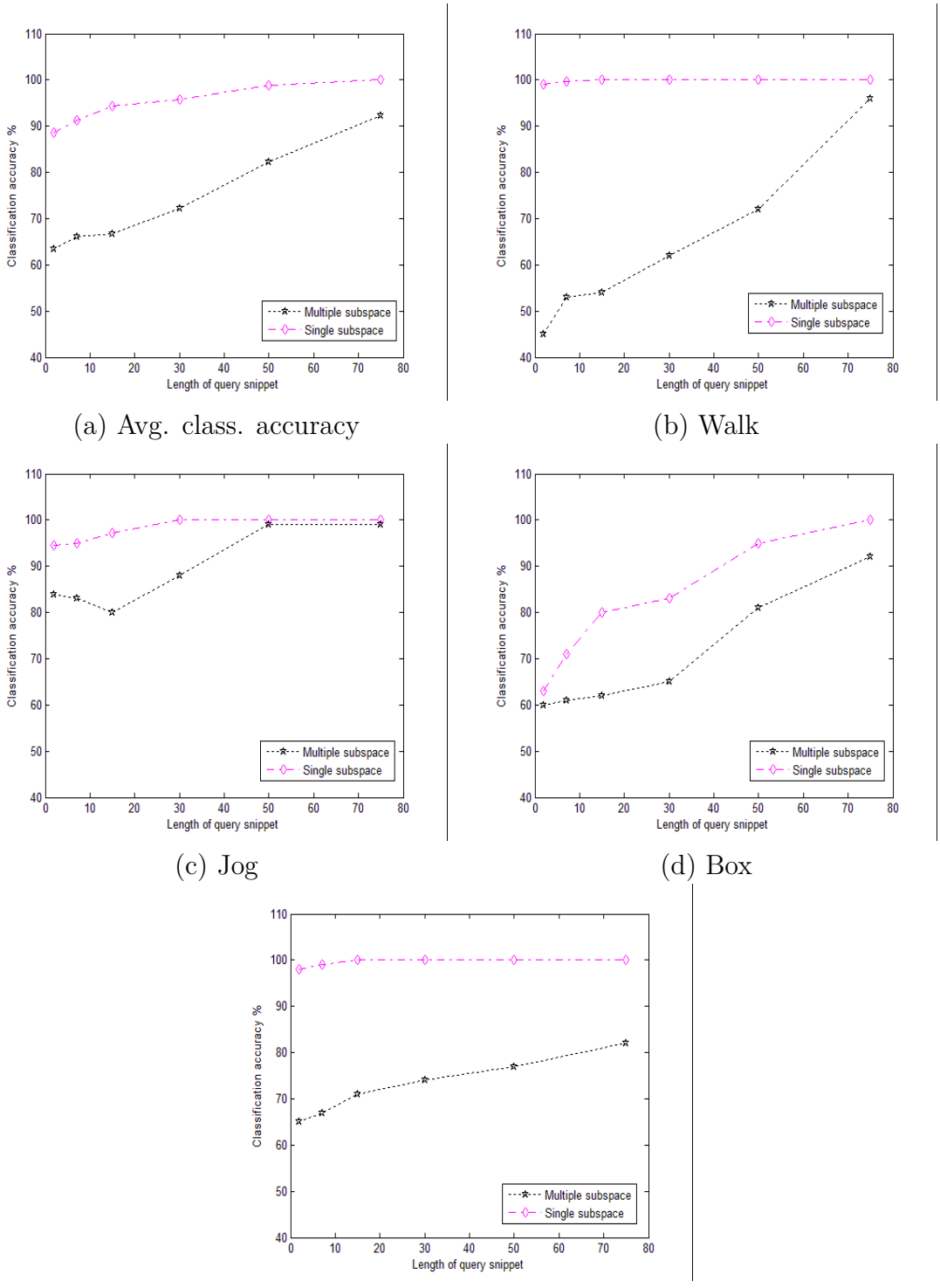
Table 5.1: Comparison of classification accuracy on the HumanEva dataset

| Algorithm | Walk | Jog | Box | Gestures | Average |
|---|---|---|---|---|---|
| Ning et al. | 100 | 100 | 98.5 | 79.1 | 95.2 |
| Single subspace | 100 | 100 | 100 | 100 | 100 |
| Multiple subspace | 97.8 | 100 | 93.2 | 83.4 | 93.5 |

snippets of the multiple subspace framework is comparable to the classification rate reported by Ning et al. [81], while we achieve 100% classification accuracy with the single subspace framework. It is worth noting that Ning et al. [81] represent the feature vectors in a bag-of-words framework, using a $300 - bin$ histogram for each frame in a 7-frame test/training feature vector sequences, amounting to a 2100D feature vector. Our feature vector, for comparison, has 298 numbers only ($\mathbf{x} : 75 \times 2, \mathbf{s} : 74 \times 2$).

*Zsolt et al.* [47]. We also compare our algorithm with Zsolt et al. [47], with 17D full-body joint angles in sequences of varying lengths (action primitives), the maximum length being 25 frames (425 dim). Zsolt et al. [47] report their classification results for HumanEva dataset, with 95% accuracy for $S1$ *walk* sequence, while we obtain better classification accuracy at 98.1% for $S1$ *walk* sequence with multiple subspace and 100% accuracy with single subspace system, compared with average walk accuracy of 97.8% (Table 5.1).

**Minimum Length of Snippets.** An important consideration for classifying human actions are the required length of action snippets for accurate classification. For image-based high-dim classification framework [103], show that a length of 5 frames (snippets) is required for accurate classification, and Ning et al. ([81]), report their classification framework using a 7-frame action sequence for image-based subspace classification framework. Similar to the work by Schindler et al. [103], we attempt to identify the minimum length of snippets to accurately classify action for skeletal features-based subspace classification framework. We use the classification accuracy reported by Ning et al [81] as the baseline in our experiment to identify the minimum action length, and based on our experi-




(a) Avg. class. accuracy

(b) Walk

(c) Jog

(d) Box

(e) Gestures action

Figure 5.8: Comparison of classification accuracy for multiple subspace and single subspace classification system for HumanEva dataset, with varying length of query snippets

mental results, the minimum length for comparable accuracy for single subspace framework is 8 frames (Table 5.2 (b)) and obtain better results from 30 frames (Table 5.2 (d-e)). The multiple subspace framework, on the other hand, reports comparable accuracy for 75-frame snippets (Table 5.2(e)). We report our classification results, as confusion matrices, in Table 5.2 and Figure 5.8. The confusion matrices for single subspace and multiple subspaces are shown in 5.4 and 5.3.

#### 5.5.1.3 CMU Motion Capture Experiments (Dataset and Algorithm Parameters)

**Dataset.** We use motion capture data of $walk, run$ and $jump$ performed by different subjects from the CMU mocap database [27]. Each observation is a 56D vector of joint angles that characterise the pose. The original 120 Hz framerate is subsampled by 4 to obtain mocap data by 30Hz framerate. This particular set of actions was chosen, so we could compare the performance of our algorithms with the result reported in a recent state-of-the-art classification algorithm [109].

**Training Parameters.** We used the same training parameters as for the HumanEva data set. The CMU data set uses 56D joint angle poses. The training and test data sets consist, for each action, of 3 to 5 subjects, 500-800 training frames, and $100 - 500$ test query snippets. The test data set contains new subjects, not present in the training data set.

#### 5.5.1.4 CMU Motion Capture Experiments (Results)

**Classification Accuracy.** We compare our classification accuracy with the results reported in Shyr et al. [109] for the same set of actions, i.e., $walk, run$ and $jump$. Our classification accuracy for the 75-frame action snippets as shown in Table 5.5, is either better (single subspace) or comparable (multiple) to performance of Shyr et al. [109] algorithm for multi-frame snippets. Additionally they also report the classification accuracy for a few other algorithm, which we report

Table 5.2: HumanEva dataset classification accuracy for varying query lengths

| Algorithm | Walk | Jog | Box | Gestures | Avg |
|---|---|---|---|---|---|
| Ning et al. | 100 | 100 | 98.5 | 79.1 | 95.2 |
| Single subspace | 99 | 94.5 | 63 | 98 | 88.6 |
| Multiple subspace | 45 | 80 | 44 | 65 | 58.3 |

(a) 3-frame

| Algorithm | Walk | Jog | Box | Gestures | Avg |
|---|---|---|---|---|---|
| Ning et al. | 100 | 100 | 98.5 | 79.1 | 95.2 |
| Single subspace | 100 | 97.2 | 80 | 100 | 94.3 |
| Multiple subspace | 54 | 84 | 45 | 71 | 63.5 |

(c) 16 frames

| Algorithm | Walk | Jog | Box | Gestures | Avg |
|---|---|---|---|---|---|
| Ning et al. | 100 | 100 | 98.5 | 79.1 | 95.2 |
| Single subspace | 99.7 | 95 | 76.3 | 99 | 93.1 |
| Multiple subspace | 53 | 83 | 42 | 67 | 61.3 |

(b) 8-frame

| Algorithm | Walk | Jog | Box | Gestures | Avg |
|---|---|---|---|---|---|
| Ning et al. | 100 | 100 | 98.5 | 79.1 | 95.2 |
| Single subspace | 100 | 100 | 83 | 100 | 95.6 |
| Multiple subspace | 63 | 88 | 58 | 74 | 71.7 |

(d) 30-frames

| Algorithm | Walk | Jog | Box | Gestures | Average |
|---|---|---|---|---|---|
| Ning et al. | 100 | 100 | 98.5 | 79.1 | 95.2 |
| Single subspace | 100 | 100 | 95 | 100 | 98.7 |
| Multiple subspace | 72 | 99 | 66 | 77 | 79 |

(e) 50-frames

Table 5.3: HumanEva confusion matrices for different query snippets lengths(multiple subspaces)

| | Walk | Jog | Box | Gestures |
|---|---|---|---|---|
| Walk | 72 | 28 | 0 | 0 |
| Jog | 1 | 99 | 0 | 0 |
| Box | 0 | 0 | 81 | 19 |
| Gestures | 0 | 0 | 23 | 77 |

(a) 30-frames

| | Walk | Jog | Box | Gestures |
|---|---|---|---|---|
| Walk | 62 | 36 | 0 | 2 |
| Jog | 1 | 88 | 3 | 8 |
| Box | 0 | 8 | 64 | 19 |
| Gestures | 0 | 12 | 43 | 45 |

(b) 50-frames

| | Walk | Jog | Box | Gestures |
|---|---|---|---|---|
| Walk | 97.8 | 2.2 | 0 | 0 |
| Jog | 0 | 100 | 0 | 0 |
| Box | 0 | 0 | 93.2 | 6.8 |
| Gestures | 0 | 1 | 15.6 | 83.4 |

(c) 75-frames

Table 5.4: HumanEva confusion matrices for different query snippets lengths(single subspace)

| | Walk | Jog | Box | Gestures |
|---|---|---|---|---|
| Walk | 100 | 0 | 0 | 0 |
| Jog | 0 | 100 | 0 | 0 |
| Box | 0 | 0 | 83 | 17 |
| Gestures | 0 | 0 | 0.4 | 99.6 |

(a) 30-frames

| | Walk | Jog | Box | Gestures |
|---|---|---|---|---|
| Walk | 100 | 0 | 0 | 0 |
| Jog | 0 | 100 | 0 | 0 |
| Box | 0 | 0 | 95 | 5 |
| Gestures | 0 | 0 | 0 | 100 |

(b) 50-frames

| | Walk | Jog | Box | Gestures |
|---|---|---|---|---|
| Walk | 100 | 0 | 0 | 0 |
| Jog | 0 | 100 | 0 | 0 |
| Box | 0 | 0 | 100 | 0 |
| Gestures | 0 | 0 | 0 | 100 |

(c) 75-frames

Table 5.5: CMU dataset: comparison of classification accuracy for multi-frame motion capture sequences

| Algorithm | SDR[109] | KDR [109] | SVM [109] | PCA[109] | Single subspace | Multiple subspace |
|---|---|---|---|---|---|---|
| Class. Accuracy % | 97.1 | 96 | 95 | 92 | 98.3 | 96.3 |

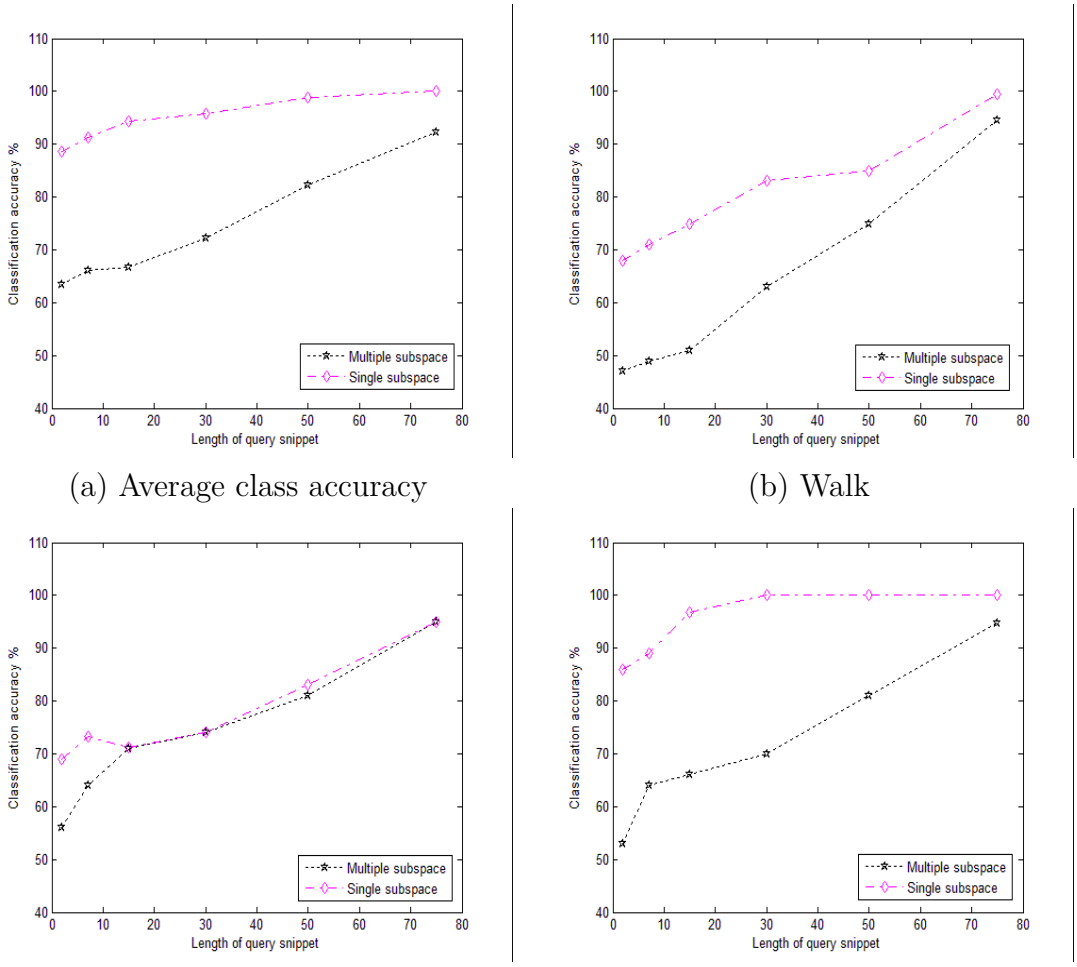
Table 5.6: CMU dataset: comparison of classification accuracy with varying query length

| Algorithm | Single subspace | Multiple subspace |
|---|---|---|
| 75-frames | 98.3 | 96.3 |
| 50-frames | 90 | 80 |
| 30-frames | 87 | 70 |
| 16-frames | 81 | 63 |
| 8-frames | 78 | 60 |
| 3-frames | 75 | 52 |

in Table 5.5. We can observe that the performance of our algorithm is better than the reported algorithms. The average classification accuracy over varying length query snippet is shown in Figure 5.9.

**Minimum Length of Snippets**. The results obtained on CMU dataset, reflect the results observed for HumanEva dataset. Specifically we obtain the best classification accuracies with 75 frame snippets as reported in Table 5.6. Similar to the results observed on HumanEva dataset, the classification accuracy of the multiple subspace framework degrades substantially compared with the single subspace framework. The classification accuracies across varying query snippet lengths is shown in Figure 5.9. The confusion matrices for single subspace and multiple subspace is shown in Table 5.7 and 5.8.

In this section, so far, we have reported the classification accuracies of our frameworks and compared the results obtained with comparable state-of-the-art systems, Additionally, we have also identified the minimum snippet length required for skeletal-features-based subspace human action classification frameworks. Moreover, we have demonstrated that the single subspace classification framework, consistently outperforms the multiple subspace classification framework over varying query snippet lengths. We next evaluate both the algorithms,




(a) Average class accuracy (b) Walk

(c) Run (d) Jump

Figure 5.9: CMU dataset: comparison of classification accuracy for single and multiple subspace frameworks

Table 5.7: CMU confusion matrices for different query snippets lengths(multiple subspaces)

| | Walk | Run | Jump |
|---|---|---|---|
| Walk | 63 | 35 | 2 |
| Run | 19 | 80 | 1 |
| Jump | 2 | 28 | 70 |

(a) 30-frames

| | Walk | Run | Jump |
|---|---|---|---|
| Walk | 72.1 | 18.1 | 9.8 |
| Run | 9.5 | 90.5 | 0 |
| Jump | 7 | 13.1 | 78.9 |

(b) 50-frames

| | Walk | Run | Jump |
|---|---|---|---|
| Walk | 94.5 | 5.5 | 0 |
| Run | 0 | 100 | 0 |
| Jump | 2.2 | 3 | 94.8 |

(c) 75-frames

Table 5.8: CMU confusion matrices for different query snippets lengths(single subspace)

| | Walk | Run | Jump |
|---|---|---|---|
| Walk | 83 | 15 | 2 |
| Run | 19 | 74 | 7 |
| Jump | 0 | 0 | 100 |

(a) 30-frames

| | Walk | Run | Jump |
|---|---|---|---|
| Walk | 85 | 13 | 2 |
| Run | 16 | 83 | 1 |
| Jump | 0 | 0 | 100 |

(b) 50-frames

| | Walk | Run | Jump |
|---|---|---|---|
| Walk | 99.5 | 0.5 | 0 |
| Run | 5 | 95 | 0 |
| Jump | 0 | 0 | 100 |

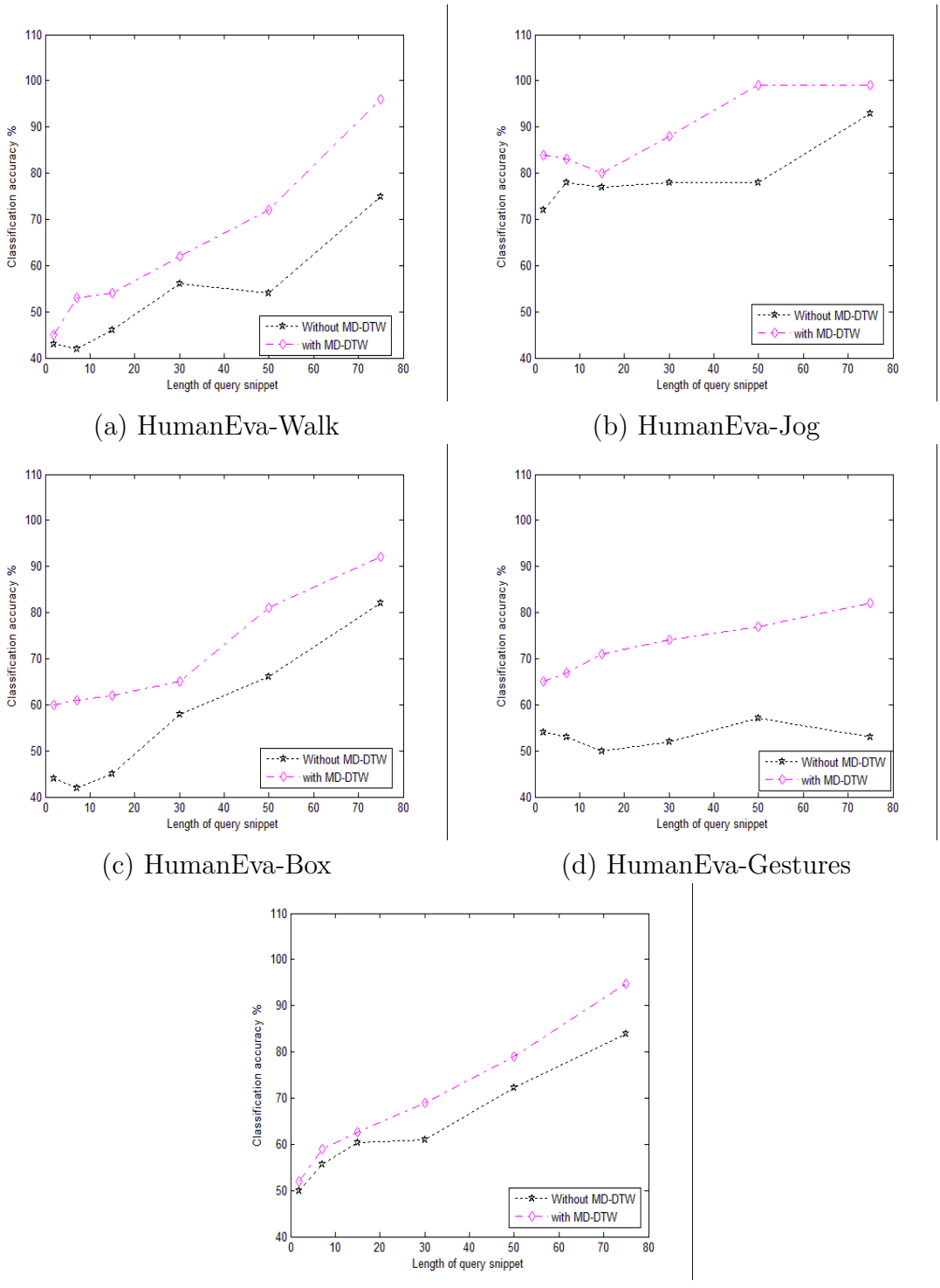
(c)75 frames

before summarising our observations.

### 5.5.2 Performance Evaluation of Multiple Subspace Classification

#### 5.5.2.1 Distance Measure Evaluation

The MDDTW is an important layer in our human action classification framework to classify actions with subtle variations, in our case walk-jog (HumanEva) and walk-run (CMU). We test the effectiveness of using MDDTW, by comparing the results of 4 layered scheme with a 3 layered scheme, obtaining the output class label with Euclidean distance measure alone, i.e. the action class $c$, whose candidate set $U^c$ gives the smallest Euclidean distance. The comparison results are shown in Figure 5.10 and Figure 5.11 where using MDDTW significantly improves the classification accuracy for both HumanEva and CMU dataset con-




(a) HumanEva-Walk

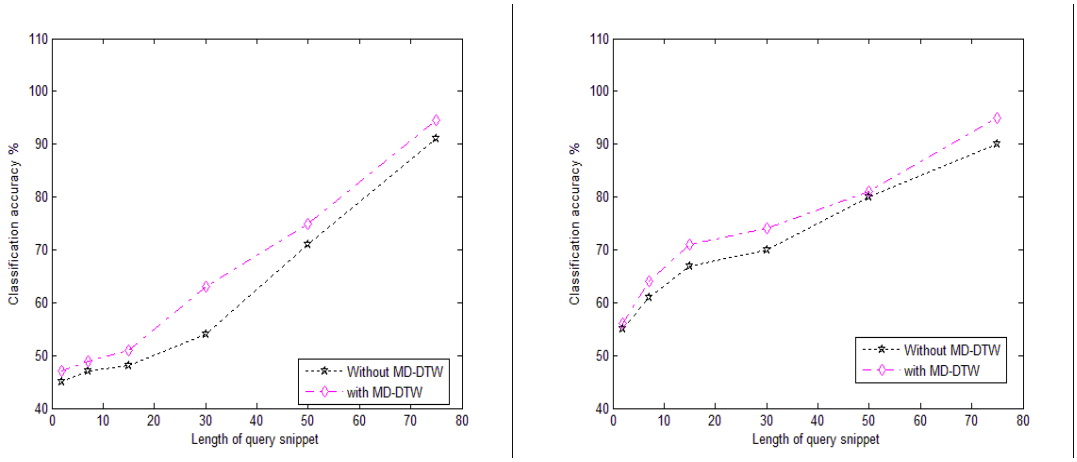
(b) HumanEva-Jog

(c) HumanEva-Box

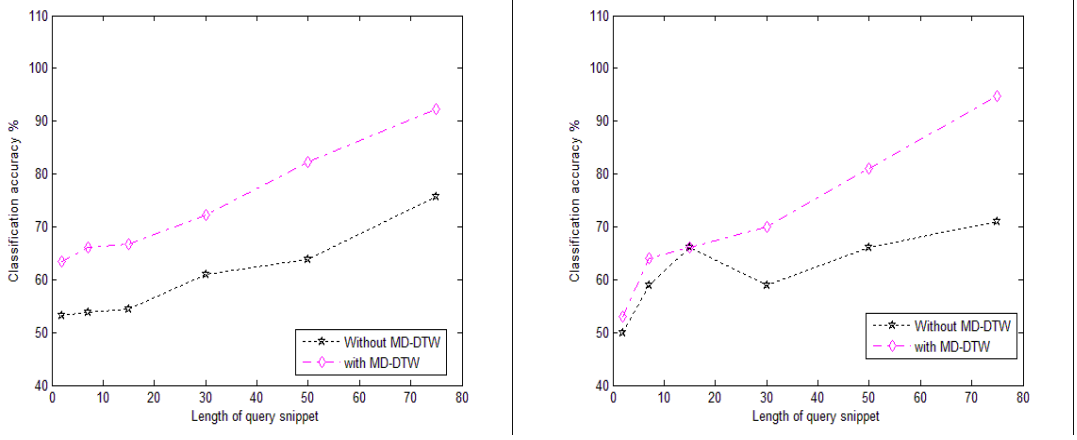
(d) HumanEva-Gestures

(h) Avg HumanEva

Figure 5.10: Comparison of classification accuracy for multiple subspace framework, on HumanEva dataset, with and without MDDTW

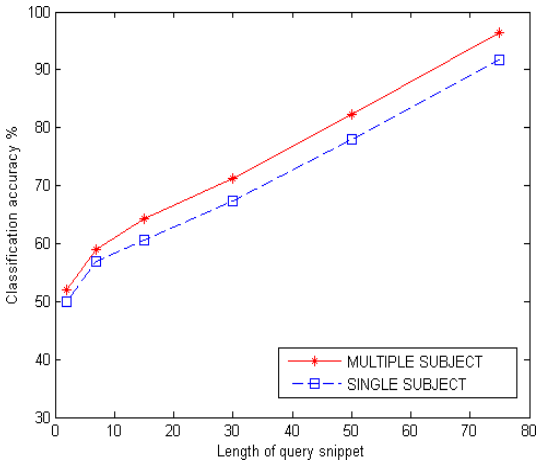



(e) CMU-Walk (f) CMU-Run




(g) CMU-Jump (i) Avg CMU

Figure 5.11: Comparison of classification accuracy for multiple subspace framework, on CMU dataset, with and without MDDTW




(a) Multiple subject vs single subject training

Figure 5.12: CMU dataset: comparison of classification accuracy for single subject and multiple subject training (multiple subspace)

sistently across the varying query snippet lengths.

#### 5.5.2.2 Single-Subject Training

We vary the number of subjects present in the training dataset and evaluate the performance of our algorithm. In the CMU training data set, we train using one subject, and test using different subjects. The results obtained in Figure 5.15 show that performance is, for this set of actions, nearly comparable to the one when training on multiple subjects.

### 5.5.3 Performance Evaluation of Single Subspace Classification

#### 5.5.3.1 Distance Measure Evaluation

Similar to the test performed on the multiple subspace framework, we also test the effectiveness of using MDDTW by comparing the results of 3 layered scheme with a 2 layered scheme, obtaining the output class label with Euclidean distance measure alone. The comparison results are shown in Figure 5.13 and Figure 5.14,

where MDDTW improves the accuracy across varying snippet lengths.

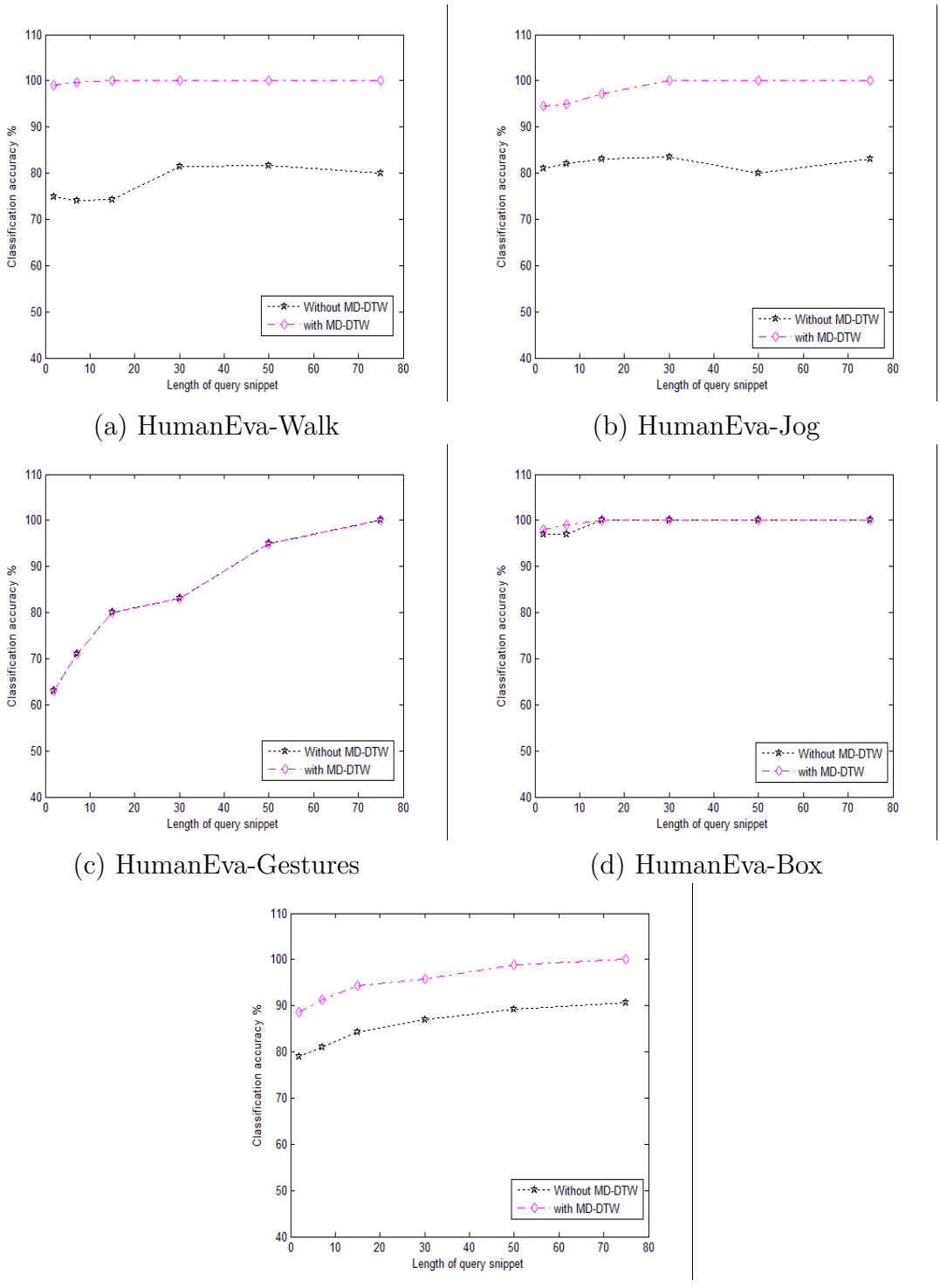
#### 5.5.3.2 Single-Subject Training

We test system performance with a single subject in the CMU training data set, testing on different subjects (multi-subject test dataset). We use queries with $100 - 300$ frames. The results obtained in Figure 5.15 show that performance is, for this set of actions, nearly comparable to the one when training on multiple subjects.

### 5.5.4 Comparative Discussion of Multiple Subspace and Single Subspace Classification

We have demonstrated that single subspace framework performs consistently better than multiple subspace framework, which can be attributed to inter-class spatial distances, which is inherent in single subspaces and absent in multiple subspaces. Specifically, as shown in Figure 5.16 (a), a discernible distance is seen across different actions in single subspace. The inter-class spatial distance is inherent to single subspace framework, as we have a single co-ordinate system for multiple actions. On the other hand, multiple subspace framework does not have a single co-ordinate system or inter-class spatial distance, as separate subspaces with independent co-ordinate systems for multiple actions are learnt.

In Figure 5.16, an example of gesture query embedding in single and multiple subspace framework is shown. As seen in Figure 5.16(a), the gesture query snippet is mapped closest to gesture partition in single subspace and the next nearest partition is box action-a similar action. In case of multiple subspace framework, we can see that gesture is mapped accurately onto both gesture and box subspace Figure 5.16(b), as they are similar actions. This mapping property, especially for similar actions, demonstrates that in multiple subspace framework, the classifica-




(a) HumanEva-Walk

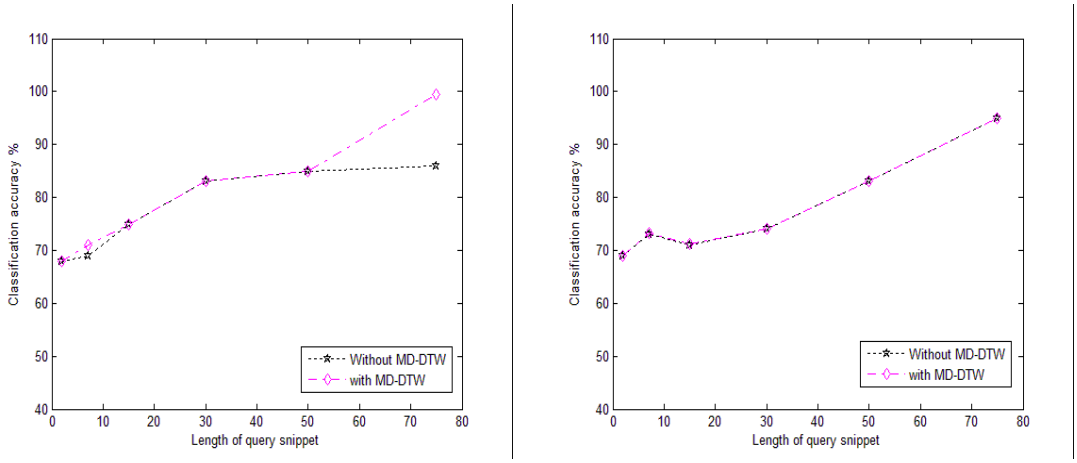
(b) HumanEva-Jog

(c) HumanEva-Gestures

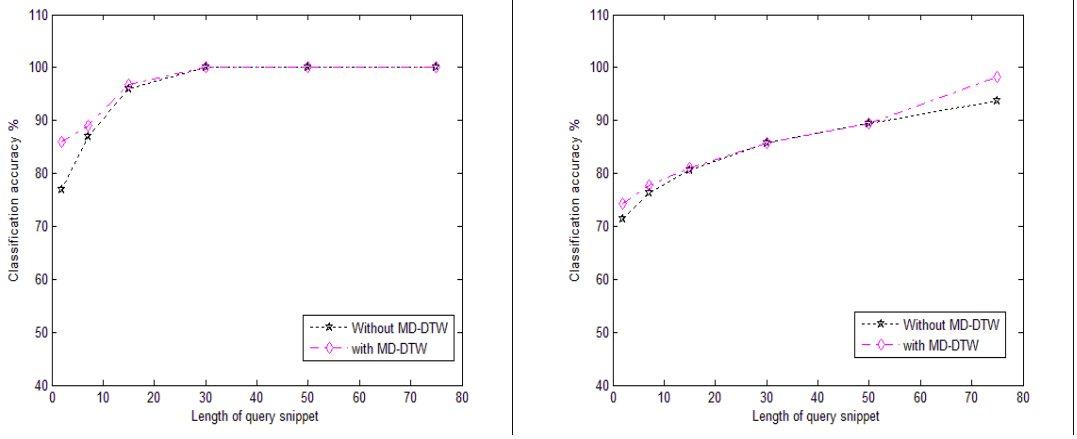
(d) HumanEva-Box

(h) HumanEva-Avg

Figure 5.13: Comparison of classification accuracy for single subspace framework, on CMU and HumanEva dataset, with and without MDDTW

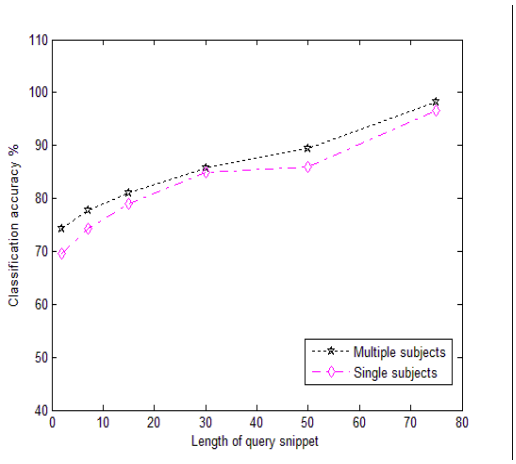



(e) CMU-Walk (f) CMU-Run




(g) CMU-Jump (i) CMU-Avg

Figure 5.14: Comparison of classification accuracy for single subspace framework, on CMU and HumanEva dataset, with and without MDDTW

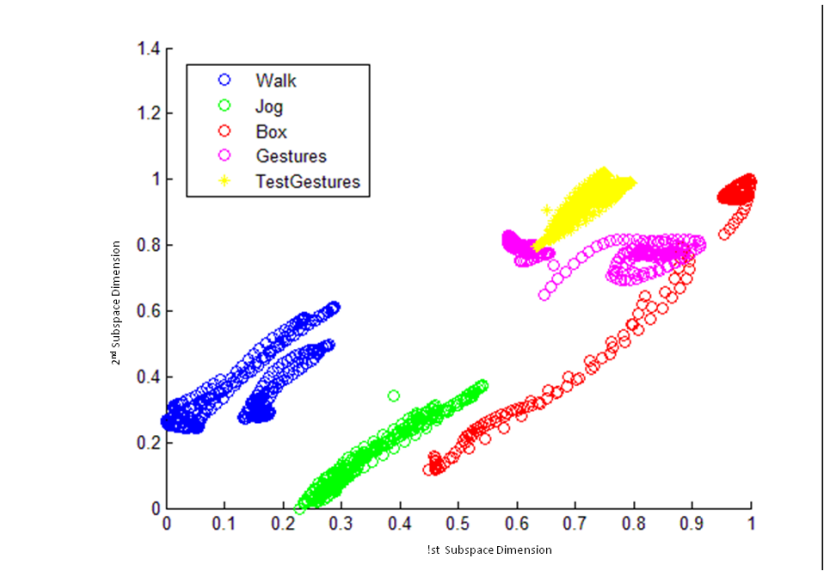



(a) Multiple subject vs single subject training

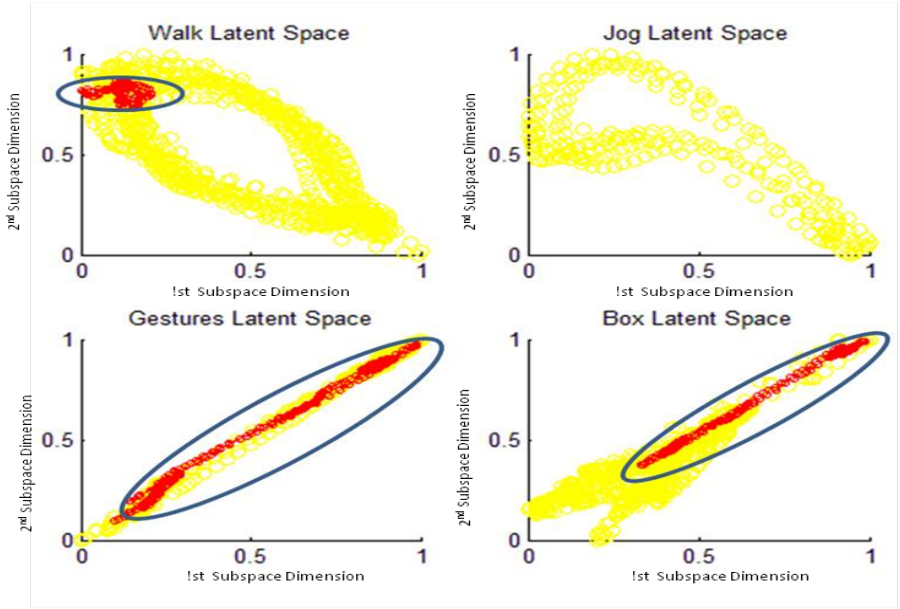
Figure 5.15: CMU dataset: comparison of classification accuracy for single subject and multiple subject training (single subspace)

tion depends greatly on mddtw measurement. While spatial distance and mddtw measurement are used for action classification in single subspace classification framework.

*Computational Time.* We report the time taken for our multi-subspace classification algorithm. Each snippet is classified with an average time of 90 sec (layer one: 87-88 sec, layer two: 0.007 to 0.02 sec, layer three: 0.06 to 0.1 sec and layer four: 0.1 to 0.3 sec), with time varying with the snippet length. It is worth noting, how computational time increases (layers 2-4) after mapping (layer 1), reflecting increased ambiguity among classes, and increasingly demanding classification search. On the other hand, in the single subspace classification framework, each snippet is classified with an average time of 23 sec (mapping query: 20-25 sec, layer one: 0.007 to 0.02 sec, layer two: 0.06 to 0.1 sec and layer three: 0.1 to 0.3 sec), with time varying with the snippet length.




(a)




(b)

Figure 5.16: Query action mapping to (a) single subspace and (b) multiple subspace

## 5.6 Conclusion and Future Work

In this chapter, we have presented a framework for markerless articulated human motion classification with multiple-view sequences. Our main contributions to the current literature, include an investigation of charting for action classification; identification of minimum snippet length required for accurate classification for

subspace skeletal features; and a comparison of our subspace classification systems with single subspace and multiple subspaces. We deploy a multi-layered classification scheme, using a a compact representation based on key-frames in subspace for action pruning of action. Our proposed tracking framework is able to classify efficiently without learning explicit models of transition or increasing the inter-class discrimination . However, we believe that incorporating a model to increase the inter-class discrimination during subspace learning, similar to Shyr et al. [109] and linear discriminant analysis, would not only improve the classification accuracy for shorter snippets, but would be useful when the number of candidate actions are high. In our current and future work, we would be concentrating on incorporating an inter-class discrimination enhancement algorithm like linear discriminant analysis within the charting framework. Additionally we would investigate switching between motion models in subspace, robust analysis of long sequences, and the application of our scheme to biomedical and animation scenarios. Finally we would like to avoid the use of threshold parameters in our system.

# Chapter 6

# Conclusion and Future Work

## 6.1 Introduction

In this chapter, we review and summarise the work presented in this thesis, highlighting the key-contributions of our research. We also identify the limitations of our system, with their possible causes, before discussing the possible extensions and future direction of our research.

*Chapter Layout.* This chapter is organised as follows. Section 6.2 summarises the work presented in our thesis. The key contributions of our work are summarised in Section 6.3. In Section 6.4 we discuss the limitations of our work and their possible causes, which form the basis for our possible extensions and potential future direction of our work in Section 6.5. Finally, we conclude our thesis in Section 6.6.

## 6.2 Summary of Thesis

In this thesis, we contribute to the existing literature of vision-based human motion analysis by introducing two human motion tracking algorithms and a human motion classification algorithm. The human motion analysis techniques described in our work are video-based, markerless, multiple-view and studio-based algorithms, and typically, marker-less human motion algorithms address the expensive and time-consuming setup associated with commercial marker-based motion capture systems, considered as state-of-the-art. We next provide a summary of our work.

*Hierarchical Particle Swarm Optimisation.* In our first markerless human motion tracking system (Chapter 3), we formulate a full-body articulated tracking from multiple-view sequences as a non-linear optimisation problem. We adopt a powerful swarm-intelligence algorithm, the particle swarm optimisation (PSO) to solve this problem. Additionally, we exploit the inherent hierarchy within the kinematic structure of the human body, to propose a hierarchical human motion tracking algorithm. Our PSO-based tracker eliminates the need for sequence-specific motion model, initialises automatically and functions as a black-box system-same algorithm with fixed parameters across different motions. However the black-box system results in increased computational complexity, which is addressed in a modified PSO-based tracker termed as the adaptive PSO-based tracker (A-PSO). APSO uses the online tracking information to adaptively vary the algorithm parameters. We perform three sets of experiments, namely comparison of HPSO and APSO with comparable state-of-the-art tracking algorithms; evaluating the effect of algorithmic parameter change; comparison between HPSO and APSO. Based on our experimental results, both our systems (HPSO and APSO) demonstrate good tracking accuracy across different actions on different datasets including HumanEva, Surrey sequences, Lee Walk and our studio dataset. The tracking performance of HPSO and APSO is comparable to existing

algorithms, while performing better on certain faster actions. Finally, we show that APSO, compared to HPSO, reduces the computational complexity, while reporting similar tracking performance.

*Charting-based Subspace Tracking.* In our second markerless human motion tracking system (Chapter 4), we exploit the prior information of motion being tracked to improve the tracking accuracy and the overall robustness of the system. We exploit the prior motion information in the form of low-dimensional subspace learnt using charting, a non-linear subspace learning technique. Tracking takes place in the learnt subspace using a modified particle swarm optimisation algorithm biased for subspace optimisation. Our proposed modified particle swarm optimisation uses motion information as a temporal and search constraint. Additionally, the computational cost associated with generative tracking's hypothesis evaluation is avoided, by using shape context histograms-based descriptors as our feature descriptors instead of multi-view silhouettes. Finally, during tracking, the subspace hypothesis are evaluated by learning the mapping from the subspace to the shape context histogram-based descriptors using multi-variate relevance vector machines (MVRVM). Similar to the experimental setup for HPSO, we perform three sets of experimental analysis on our proposed subspace tracking system. Firstly, we compare our proposed charting-based subspace tracking system with comparable state-of-the-art tracking algorithms. Secondly, we perform a computational and performance evaluation of our subspace system, specifically focusing on the modified particle swarm optimisation and subspace hypothesis evaluation. Finally, we compare our subspace tracking system with our first tracking system (HPSO) and report our observations. We perform our experiments on HumanEva and our studio dataset. In our first experimental setup, we show that the tracking performance of charting subspace tracking algorithm is better than GPAPF and a few comparable algorithms, across all actions, while reporting similar accuracies for a few actions. In the second set of experiments, we show that modified particle swarm optimisation and subspace hypothesis evaluation increases tracking per-

formance, while greatly decreasing the computational time. Finally, we show that our subspace-based tracking system performs comparably to HPSO, while greatly reducing the computational time.

*Charting-based Multi-Layered Human Action Classification.* In our third and final human motion analysis system (Chapter 5), we have presented a framework for markerless articulated human motion classification with multiple-view sequences using skeletal features obtained from either our first two systems or marker-based motion capture systems. We perform our classification, assignment of action labels to video sequences, in a low-dimensional subspace learnt using charting. We present a multi-layered classification scheme using key-frames extracted from the subspace. The main motivation of adopting a multi-layered classification scheme is the layered pruning of candidate actions with less demanding classification search till only fewer actions with subtle variations remain in the final layer. In our final layer we adopt multi-dimensional dynamic time warping, a feature vector alignment algorithm, to accurately assign the action label to the video sequence.We present two variations of subspace-based human motion classification algorithm, namely single subspace and multiple subspace multi-layered classification algorithm-the difference being the method of learning the charting-based subspaces for different actions. In the first system, a single subspace is learnt for multiple actions, providing a single subspace co-ordinate system for all actions. On the other hand, in the second system, we learn separate independent subspaces for each action, thus obtaining separate co-ordinate systems for each action. We report good classification accuracies, on the HumanEva dataset [114] and CMU motion capture dataset [27], which are comparable with the existing state-of-the-art tracking systems. Furthermore, we compare the multiple subspace and single subspace systems, and present our observations of the differences in the two algorithms.

## 6.3 Summary of Key Contributions

In this section, we summarise the key contributions of our three human motion analysis systems. We highlight the major contributions with **bold** texts, while the minor contributions are highlighted in *italics*.

- Markerless human motion tracking using particle swarm optimisation without any motion prior (Chapter 3)
  - **Particle swarm optimisation used for articulated full-body tracking.**
    - *A novel, hierarchical version of particle swarm optimisation algorithm (H-PSO) is used for full-body markerless human motion tracking.*
    - *A guiding-cylinder scheme is proposed for the hypothesis evaluation in H-PSO, providing spatial and temporal constraints and reducing the computational complexity.*
    - *An adaptive version of H-PSO is proposed, wherein the system parameters are automatically varied online based on the online accuracy of tracking, thus reducing the computational complexity.*
- Markerless human motion tracking with particle swarm optimisation using subspace learning-based motion prior (Chapter 4)
  - **Charting not previously used for subspace human motion tracking**.
  - **Particle swarm optimisation not previously used for subspace tracking**.
  - **A modified particle swarm optimisation, specific for subspace tracking called subspace PSO is proposed.**
- Markerless human motion classification with multi-layered classification framework (Chapter 5)
  - **Charting not previously used for human motion classification.**

- **Estimating the minimum length of skeletal features required for accurate human motion classification.**

  * Derivation of sequence of key-poses from the human action subspace.
  * Comparison of multiple subspace and single subspaces for human action classification.

## 6.4 Limitations and Possible Causes

In this section, we first report a few identified limitations of our system, suggest the possible causes for the weaknesses.

### 6.4.1 Hierarchical Particle Swarm Optimisation

We identify four main weaknesses in our first proposed markerless tracking framework-hierarchical particle swarm optimisation. Firstly, as our proposed system belongs to a generative tracking framework and eliminates the need for a motion model, the tracking performance depends on the quality of measurement, in our case of silhouettes, which tend to be noisy. Additionally, in the absence of motion model, tracking in the presence of occlusions is challenging. In our experiments, tracking error was present for a few frames and recovery achieved systematically after one or a few frames, for those few errors. Wrong pose estimates seem to depend mainly on poor silhouette segmentation in some cameras. Secondly, the hierarchical structure of HPSO suggests that incorrect estimates at early stages of hierarchy will affect the accuracy of estimates for subsequent limbs. Although APSO improves the tracking accuracy and addresses the problem of "error-propagation", it is still prone to wrong pose estimate in case of noisy and occluded silhouettes. Thirdly, the body model that we use, composed of cylinders [8] introduces a front-back ambiguity for poses in which all skeleton

segments lie in a plane. This problem would be solved by nonsymmetric surface models, as used by Balan et al. [9]. Finally, the computational time is high, owing to the expensive silhouette-based hypothesis evaluation.

### 6.4.2 Charting-based Subspace Tracking

In our charting-subspace-based tracking framework, we demonstrated the benefits of incorporating a motion prior in the tracking framework. However, integrating a subspace-based motion prior in the tracking, requires the initialisation of the tracking algorithm manually. Specifically, the subspace corresponding to the action being tracked needs to be selected manually. Another identified weakness is the use of 2D shape descriptors, making our subspace-based tracking system dependent on a particular camera-view and studio setup. In practical terms, the learning of subspace, shape context descriptors and mapping is specific for each studio setup, and is incompatible with a different studio setup, especially if the camera arrangements are different. This arises as a result of our shape-context histogram-based descriptors not being camera-invariant.

### 6.4.3 Charting-based Human Motion Classification

In our charting-based human motion classification, we report good classification accuracy. However, we believe that our system would not be able to identify similar actions with subtle style changes, for example, normal walking, sad walking and happy walk. Our proposed tracking framework is able to classify efficiently without learning explicit models of transition or increasing the inter-class discrimination for the number of actions we have used. However, we believe that an increase in the number of actions (>10) could result in a possible reduction of inter-class spatial distances in the single subspace, as described in Section 5.5.4, resulting in an increase in the minimum length of action snippets required.

## 6.5 Future Work

In the previous section, we identified certain weaknesses in our work and suggested possible causes, which are addressed in this section in context of our discussion of the future direction of our work.

### 6.5.1 Hierarchical Particle Swarm Optimisation

In our charting-based tracking system, we incorporated the motion prior, thus increasing the robustness of the tracker, while greatly decreasing the computational time. However, there is sufficient scope for future work on HPSO in the high-dim space itself, which would improve the tracking accuracy and reduce the computational complexity. Firstly, the front-back ambiguity can be avoided by using a complicated body model, which would increase the pose estimation accuracy to a certain degree. Secondly, a motion prior from learnt examples can be used as a prior to initalise the PSO search in every frame. Thirdly, a discriminative framework can be integrated within our generative scheme, which can be used to initialise the PSO search in every frame, this would be useful, especially for fast actions like a break dance. Finally, as ascertained from our experimental results, the tracking accuracy increases with the number of particles. However using, say, 100 particles would greatly increase the computational time, which can be avoided by using GPU-based particle swarm optimisation.

### 6.5.2 Charting-based Subspace Tracking

The manual identification of subspace before initialisation of tracking, can be eliminated by incorporating an action classification within the subspace tracking framework. In our future work, we would address the issue of tracking a multiple action video sequence by either learning a single representation, for example sub-

space, for multiple actions or by learning the transition model between different subspaces. Additionally, we would like to incorporate hierarchical subspaces (eg, hGPLVM) and subspace dynamics (eg, GPDM) which could improve the tracking accuracy. Finally, the camera variant problem of 2D shape descriptors can be avoided by using 3D shape descriptors [108] in our subspace tracking framework.

### 6.5.3 Charting-based Human Motion Classification

In our future work, we would be focusing on improving the classification accuracy and reducing the length of minimum snippets, by formulating charting as a discriminative classifier to increase the inter-class discrimination in the subspace, especially when the number of actions are high. In this regard, we would be investigating techniques like sufficient dimensionality reduction [109] and linear discriminant analysis. Additionally we would investigate switching between motion models in subspace. Finally we would like to avoid the use of threshold parameters in our system.

## 6.6 Concluding Remarks

Given the current state-of-the-art and advances in human motion analysis, we believe that many challenges present in markerless human motion analysis will be met. This would meet the requirements of an ideal video-based human motion analysis system, capable of robustly and accurately extracting human motion information from any video sequence, potentially captured in a wide-range of environments.

# Bibliography


[1] A. Agarwal and B. Triggs. Tracking articulated motion using a mixture of autoregressive models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2004.

[2] A. Agarwal and B. Triggs. Monocular human motion capture with a mixture of regressors. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[3] A. Agarwal and B. Triggs. Recovering 3d human pose from monocular images. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 28, 2006.

[4] M. Andriluka and S. Schiele. Monocular 3d pose estimation and tracking by detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[5] L. Anton-Canalis, M. Hernandez-Tejera, and E. Sanchez-Nielsen. Particle swarms as video sequence inhabitants for object tracking in computer vision. In *Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA)*, 2006.

[6] M. Arulampalam, S. Maskell, and N. Gordon. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. In *IEEE Transactions on Signal Processing*, volume 50, 2002.

[7] Ascension. http://www.ascension-tech.com/.

[8] A. Balan, L. Sigal, and M. Black. A quantitative evaluation of video-based 3d person tracking. In *Proceedings of the International Conference on Computer Communications and Networks (ICCCN)*, 2005.

[9] A. O. Balan, L. Sigal, M. J. Black, J. E. Davis, and H. W. Haussecker. Detailed human shape and pose from images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2007.

[10] J. Bandouch, F. Engstler, and M. Beetz. Evaluation of hierarchical sampling strategies in 3d human pose estimation. In *Proceedings of British Machine Vision Conference (BMVC)*, 2008.

[11] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, 2001.

[12] A. Bissacco, M. Yang, and S. Soatto. Fast human pose estimation using appearance and motion via multi-dimensional boosting regression. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[13] J. Blackburn and E. Ribeiro. Human motion recognition using isomap and dynamic time warping. In *Proceedings of the 2nd conference on Human motion: understanding, modeling, capture and animation*, 2007.

[14] A. Bobick. Movement, activity, and action: The role of knowledge in the perception of motion. In *Royal Society Workshop on Knowledge-based Vision in Man and Machine*, volume 352, 1997.

[15] Matthew Brand. Charting a manifold. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[16] C. Bregler, J. Malik, and K. Pullen. Twist based acquisition and tracking of animal and human kinematics. In *International Journal of Computer Vision (IJCV)*, volume 56, 2004.

[17] M. Bruebacker, D. Fleet, and A. Hertzmann. Physics-based person tracking using simplified lower-body dynamics. In *International Conference of Computer Vision and Pattern Recognition (CVPR)*, 2007.

[18] B. Brumitt, B. Meyers, and J. Krumm. Easyliving: Technologies for intelligent environments. In *Second International Symposium on Handheld and Ubiquitous Computing*, 2000.

[19] F. Caillette, A. Galata, and T. Howard. Real-time 3-d human body tracking using learnt models of behaviour. In *Computer Vision and Image Understanding (CVIU)*, volume 109, 2008.

[20] J. Chen, M. Kim, Y. Wang, and Q. Ji. Switching gaussian process dynamic models for simultaneous composite motion tracking and recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, 2009.

[21] K. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette across time: Part ii: Applications to human modeling and markerless motion tracking. In *International Journal of Computer Vision*, volume 63, August 2005.

[22] K. Choo and D. Fleet. People tracking using hybrid monte carlo filtering. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2001.

[23] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[24] Contemplas. http://www.contemplas.com.

[25] E. Boyer D. Weinland and R. Ronfard. Action recognition from arbitrary views using 3d exemplars. In *International Conference on Computer Vision (ICCV)*, 2007.

[26] J. Darby, B. Li, N. Costen, D. Fleet, and N. Lawrence. Backing off: Hierarchical decomposition of activity for 3d novel pose recovery. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2009.

[27] CMU Motion Capture dataset. http://www.mocap.cs.cmu.edu.

[28] E. de Aguiar, C. Theobalt, C. Stoll, and H. Seidel. Markerless deformable mesh tracking for human shape and motion capture. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[29] H. Dee and D. Hogg. Detecting inexplicable behaviour. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2004.

[30] J. Deutscher and I. Reid. Articulated body motion capture by stochastic search. In *International Journal of Computer Vision (IJCV))*, volume 61, 2005.

[31] R. Eberhart and Y. Shi. Comparison between genetic algorithms and particle swarm optimization. In *Proceedings of the International Conference on Evolutionary Programming*, 1998.

[32] A. Elgammal and C. Lee. Nonlinear manifold learning for dynamic shape and dynamic appearance. In *Computer Vision and Image Understanding (CVIU)*, volume 106, 2007.

[33] A. Elgammal and C. Lee. Tracking people on a torus. In *Pattern Analysis and Machine Intelligence*, volume 31, March 2009.

[34] A. Fathi and G. Mori. Human pose estimation using motion exemplars. In *International Conference on Computer Vision (ICCV)*, 2007.

[35] J. Gall, B. Rosenhan, T. Brox, and H. Seidel. Optimization and filtering for human motion capture. In *International Journal of Computer Vision*, volume 87, March 2010.

[36] J. Gall, A. Yao, and L. Van Gool. 2d action recognition serves 3d human pose estimation. In *Proceedings of European Conference of Computer Vision (ECCV)*, 2010.

[37] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *International Conference on Computer Vision (ICCV)*, 2005.

[38] F. Guo and G. Qian. Learning and inference of 3d human poses from gaussian mixture modeled silhouettes. In *International Conference on Pattern Recognition*, 2006.

[39] F. Guo and G. Qian. Monocular 3d tracking of articulated human motion in silhouette and pose manifolds. In *EURASIP Journal on Image and Video Processing*, 2008.

[40] F. Guo and G. Qian. Multi-view tracking of articulated human motion in silhouette and pose manifolds. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2009.

[41] E. Gur, Y. Weizman, P. Perdu, and Z. Zalevsky. Radon-transform-based image enhancement for microelectronic chip inspection. In *IEEE Transactions on Device and Materials Reliability*, volume 10, 2010.

[42] T. Han, H. Ning, and T. Huang. Efficient nonparametric belief propagation with application to articulated body tracking. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[43] L. Herda, R. Urtasun, and P. Fua. Hierarchical implicit surface joint limits for human body tracking. In *Computer Vision Image Understanding*, volume 99, 2005.

[44] M. Hofmann and D.M. Gavrila. Multi-view 3d human pose estimation combining single-frame recovery, temporal integration and model adaptation. In *Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2009.

[45] S. Hou, A. Galata, and F. Caillette. Real-time body tracking using a gaussian process latent variable model. In *International Conference of Computer Vision (ICCV), year = 2007,*.

[46] N. Howe. Silhouette lookup for monocular 3d pose tracking. In *Image Vision Computing*, volume 25, March 2007.

[47] Z. Husz, A. Wallace, and P. Green. Human activity recognition with action primitives. In *IEEE International Conference on Advanced Video and Signal-based Surveillance (AVSS)*, 2007.

[48] Z. Husz, A. Wallace, and P. Green. Evaluation of a hierarchical partitioned particle filter with action primitives. In *CVPR 2nd Workshop on Evaluation of Articulated Human Motion and Pose Estimation*, 2007.

[49] Z. Husz, A. Wallace, and P. Green. Evaluation of a hierarchical partitioned particle filter with action primitives. In *CVPR 2nd Workshop on Evaluation of Articulated Human Motion and Pose Estimation (EHUM2)*, 2007.

[50] S. Ivekovic and E. Trucco. Human body pose estimation with pso. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, pages 1256–1263, 2006.

[51] S. Ivekovic, E. Trucco, and Y. Petillot. Human body pose estimation with particle swarm optimisation. In *Evolutionary Computation (2008)*, volume 16, 2008.

[52] T. Jaeggli, E. Koller-Meier, and L. Van Gool. Multi-activity tracking in lle body pose space. In *ICCV 2nd Workshop on HUMAN MOTION Understanding, Modeling, Capture and Animation*, October 2007.

[53] K. Jia and D.Y. Yeung. Human action recognition using local spatio-temporal discriminant embedding. In *In Computer Vision and Pattern Recognition (CVPR)*, 2008.

[54] I. Kakadiaris and D. Metaxas. Three-dimensional human body model acquisition from multiple views. In *International Journal of Computer Vision (IJCV)*, volume 30, 1998.

[55] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, volume 4, 1995.

[56] T. Kobayashi, K. Nakagawa, and J.; Guisheng Zhai Imae. Real time object tracking on video image sequence using particle swarm optimization. In *Proceedings of International Conference on Control, Automation and Systems*, 2007.

[57] W. Liang G. Hou L. Han, X. Wu and Y. Jia. Discriminative human action recognition in the learned hierarchical manifold space. In *Image Vision Computing*, volume 28, May 2010.

[58] E. Rivlin L. Raskin and M. Rudzsky. Tracking and classifying of human motions with gaussian process annealed particle filter. In *Asian Conference on Computer Vision*, 2007.

[59] N. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Neural Information Processing (NIPS)*, page 2004, 2003.

[60] Neil D. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. In *Journal of Machine Learning Research*, volume 6, 2005.

[61] R. Li, M. Yang, S. Sclaroff, and T. Tian. Monocular tracking of 3D human motion with a coordinated mixture of factor analyzers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2006.

[62] W. Li, Z. Zhang, and Z. Liu. Action recognition based on a bag of 3d points. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2010.

[63] Liberty. http://www.polhemus.com/.

[64] Z. Lin, Z. Jiang, and L. Davis. Recognizing actions by shape-motion prototype trees. In *International Conference on Computer Vision (ICCV)*, 2009.

[65] G. Liu, X. Tang, J. Huang, J. Liu, and Da Sun. Hierarchical model-based human motion tracking via unscented kalman filter. In *International Conference on Computer Vision (ICCV)*, 2007.

[66] F. Lv and R. Nevatia. Recognition and segmentation of 3d human action using hmm and multi-class adaboost. In *European Conference of Computer Vision (ECCV)*, 2006.

[67] J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2000.

[68] Marlbrook. http://www.marlbrook.com/sport.html.

[69] MetaMotion. http://www.metamotion.com.

[70] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman. Human body model acquisition and tracking using voxel data. In *International Journal of Computer Vision*, volume 53, 2003.

[71] K. Mikolajczyk and H. Uemura. Action recognition with motion-appearance vocabulary forest. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[72] L. Mündermann, S. Corazza, and T. Andriacchi. Accurately measuring human movement using articulated icp with soft-joint constraints and a repository of articulated models. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[73] T. Moeslund and E. Granum. A survey of computer vision-based human motion capture. In *Computer Vision and Image Understanding (CVIU)*, volume 81, 2001.

[74] T. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. In *Computer Vision and Image Understanding (CVIU)*, volume 104, November 2006.

[75] G. Mori and J. Malik. Recovering 3d human body configurations using shape contexts. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 28, July 2006.

[76] MotionBuilder. http://www.metamotion.com/software/motion-builder.

[77] P. Natarajan and R. Nevatia. View and scale invariant action recognition using multiview shape-flow models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[78] J. Niebles, H. Wang, and L. Fei-fei. Unsupervised learning of human action categories using spatial-temporal words. In *Proceedings of British Machine Vision Conference (BMVC)*, 2006.

[79] H. Ning, T. Tan, L. Wang, and W. Hu. People tracking based on motion model and motion constraints with automatic initialization. In *Pattern Recognition*, volume 37, July 2004.

[80] H. Ning, W. Xu, Y. Gong, and T. Huang. Discriminative learning of visual words for 3d human pose estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[81] H. Ning, W. Xu, Y. Gong, and T. Huang. Latent pose estimator for continuous action recognition. In *Proceedings on European Conference on Computer Vision (ECCV)*, 2008.

[82] Kishino F Ohya, J. Human posture estimation from multiple images using genetic algorithm. In *International Conference of Pattern Recognition*, volume 1, 1994.

[83] R. Okada and S. Soatto. Relevant feature selection for human pose estimation and localization in cluttered images. In *European Conference on Computer Vision (ECCV)*, 2000.

[84] V. Pavlović, J. Rehg, T. Cham, and K. Murphy. A dynamic bayesian network approach to figure tracking using learned dynamic models. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 1999.

[85] P. Peursum, S. Venkatesh, and G. West. Tracking-as-recognition for articulated full-body human motion analysis. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[86] Physilog. http://lmam.ep.ch.

[87] R. Poli. An analysis of publications on particle swarm optimisation applications. Technical Report CSM-649, University of Essex, Department of Computer Science, November 2007.

[88] R. Poli. Mean and variance of the sampling distribution of particle swarm optimizers during stagnation. In *IEEE Transactions on Evolutionary Computation*, volume 13, 2009.

[89] R. Poppe. Vision-based human motion analysis: An overview. In *Computer Vision and Image Understanding (CVIU)*, volume 108, 2007.

[90] R. Poppe. A survey on vision-based human action recognition. In *Image and Vision Computing*, volume 28, 2010.

[91] R. Poppe and M. Poel. Comparison of silhouette shape descriptors for example-based human pose recovery. In *International Conference on Automatic Face and Gesture Recognition*, 2006.

[92] M. Pupilli and A. Calway. Real-time camera tracking using a particle filter. In *Proceedings of the British Machine Vision Conference (BMVC)*, September 2005.

[93] Qualisys. http://www.qualisys.com/.

[94] D. Ramanan and C. Sminchisescu. Training deformable models for localization. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 206–213, New York, NY, June 2006.

[95] L. Raskin, E. Rivlin, and M. Rudzsky. Using gaussian processes for human tracking and action classification. In *Proc ISVC 2007*, 2007.

[96] L. Raskin, M. Rudzsky, and E. Rivlin. 3d human body-part tracking and action classification using a hierarchical body model. In *Proceedings of British Machine Vision Conference (BMVC)*, 2009.

[97] L. Raskin, M. Rudzsky, and E. Rivlin. Dimensionality reduction using a gaussian process annealed particle filter for tracking and classification of articulated body motions. In *Computer Vision and Image Understanding*, volume In Press, Accepted Manuscript, 2011.

[98] T. Roberts, S. McKenna, and I. Ricketts. Human tracking using 3d surface colour distributions. In *Image and Vision Computing*, 2003.

[99] C. Robertson and E. Trucco. Human body posture via hierarchical evolutionary optimization. In *Proceedings of British Machine Vision Conference (BMVC06)*, 2006.

[100] B. Rosenhahn, T. Brox, and H. Seidel. Scaled motion dynamics for markerless motion capture. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[101] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. In *SCIENCE*, volume 290, 2000.

[102] S. Roweis, L. Saul, and G. Hinton. Global coordination of local linear models. In *Neural Information Processing Systems (NIPS)*, 2001.

[103] K. Schindler and L. van Gool. Action snippets: How many frames does human action recognition require? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[104] S. Sedai, M. Bennamoun, and D. Huynh. Localized fusion of shape and appearance features for 3d human pose estimation. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2010.

[105] A. Shaji, B. Siddiquie, S. Chandran, and D. Suter. Human pose extraction from monocular videos using constrained non-rigid factorization. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2008.

[106] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *International Conference on Computer Vision (ICCV)*, 2003.

[107] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Proceedings of the IEEE International Conference on Evolutionary Computation (CEC)*, 1998.

[108] P. Shilane and T. Funkhouser. Selecting distinctive 3D shape descriptors for similarity retrieval. In *Shape Modeling International*, 2006.

[109] Urtasun R. Jordan M. Shyr, A. Sufficient dimension reduction for visual sequence classification. In *Proceedings of Computer Vision and Pattern Recognition, (CVPR)*, 2010.

[110] H. Sidenbladh, M. Black, and D. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2000.

[111] L. Sigal and M. Black. Predicting 3D people from 2D pictures. In *Proceedings of the International Conference on Articulated Motion and Deformable Objects (AMDO)*, 2006.

[112] L. Sigal, M. Isard, B. Sigelman, and M. Black. Attractive people: Assembling loose-limbed models using non-parametric belief propagation. In *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2003.

[113] L. Sigal, A. Balan, and M. Black. Combined discriminative and generative articulated pose and non-rigid shape estimation. In *Proceedings of Neural Information Processing Systems Conference, (NIPS)*, 2007.

[114] L. Sigal, A.. Balan, and M. Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. In *International Journal of Computer Vision (IJCV)*, volume 87, 2010.

[115] C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling. In *International Journal of Robotic Research*, volume 22, 2003.

[116] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Discriminative density propagation for 3D human motion estimation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[117] C. Sminchisescu, A. Kanaujia, Zhiguo Li, and D. Metaxas. Conditional models for contextual human motion recognition. In *International Conference on Computer Vision (ICCV)*, 2005.

[118] C. Sminchisescu, A. Kanaujia, and D. Metaxas. Learning joint top-down and bottom-up processes for 3D visual inference. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[119] J. Starck and A. Hilton. Surface capture for performance-based animation. In *IEEE Computer Graphics and Applications*, volume 27(3), 2007.

[120] J. Sullivan and S. Carlsson. Recognizing and tracking human action. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2002.

[121] Y. Sun, M. Bray, A. Thayananthan, B. Yuan, and P. Torr. Regression-based human motion capture from voxel data. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2006.

[122] K. Schindler D. Suter T. Chin, L. Wang. Extrapolating learned manifolds for human activity recognition. In *Proceedings of the International Conference on Image Processing (ICIP)*, 2007.

[123] G. ten Holt, M. Reinders, and E. Hendriks. Multi-dimensional dynamic time warping for gesture recognition. In *Thirteenth annual conference of the Advanced School for Computing and Imaging*, 2007.

[124] J. Tenenbaum, V. Silvj, and V. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. In *Science*, volume 290, 2000.

[125] A. Thayananthan, R. Navaratnam, B. Stenger, P. Torr, and R. Cipolla. Pose estimation and tracking using multivariate regression. In *Pattern Recognition Letters*, volume 29, 2008.

[126] M. Tipping. Sparse bayesian learning and the relevance vector machine. In *Journal of Machine Learning Research*, volume 1, 2001.

[127] M. Tipping and C. Bishop. Probabilistic principal component analysis. In *Journal of the Royal Statistical Society*, volume 61, 1999.

[128] K. Toyama and A. Blake. Probabilistic tracking with exemplars in a metric space. In *International Journal of Computer Vision*, volume 48, June 2002.

[129] D. Tweed and A. Calway. Tracking many objects using subordinated condensation. In *Proceedings of the British Machine Vision Conference (BMVC)*, October 2002.

[130] N. Ukita, M. Hirai, and M. Kidode. Complex volume and pose tracking with probabilistic dynamical models and visual hull constraints. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009.

[131] R. Urtasun. Motion models for robust 3d human body tracking. In *PhD Thesis:École Polytechnique Federale De Lausanne*, 2006.

[132] R. Urtasun and T. Darrell. Discriminative gaussian process latent variable model for classification. In *Proceedings of the international conference on Machine learning*, 2007.

[133] R. Urtasun and T. Darrell. Sparse probabilistic regression for activity-independent human pose inference. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[134] R. Urtasun and P. Fua. 3d human body tracking using deterministic temporal motion models. In *In European Conference on Computer Vision (ECCV)*, 2004.

[135] R. Urtasun, D. Fleet, and P. Fua. Monocular 3-d tracking of the golf swing. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[136] R. Urtasun, D. Fleet, A. Hertzmann, and P. Fua. Priors for people tracking from small training sets. In *Proceedings of the International Conference On Computer Vision (ICCV)*, 2005.

[137] R. Urtasun, D. Fleet, and P. Fua. 3D people tracking with gaussian process dynamical models. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[138] R. Urtasun, D. Fleet, and N. Lawrence. Modeling human locomotion with topologically constrained latent variable models. In *In Human Motion: Understanding, Modeling, Capture and Animation (HUMO)*, 2007.

[139] Vicon. http://www.vicon.com.

[140] J. Wang, D.Fleet, and A. Hertzmann. Gaussian process dynamical models. In *Advances in Neural Information Processing (NIPS)*, 2005.

[141] L. Wang and D. Suter. Learning and matching of dynamic shape manifolds for human action recognition. In *IEEE Transactions on Image Processing*, 2007.

[142] P. Wang and J. Rehg. A modular approach to the analysis and evaluation of particle filters for figure tracking. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2006.

[143] Y. Wang, K. Huang, and T. Tan. Human activity recognition based on r transform. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[144] D. Weinland and E. Boyer. Action recognition using exemplar-based embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[145] D. Weinland, R. Ronfard, and E. Boyer. Free viewpoint action recognition using motion history volumes. In *Computer Vision Image Understanding (CVIU)*, volume 104, November 2006.

[146] X. Zhang, W. Hu, S. Maybank, X. Li, and M. Zhu. Sequential particle swarm optimization for visual tracking. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[147] Liu Y. Zhao, X. Generative tracking of 3d human motion by hierarchical annealed genetic algorithm. In *Pattern Recognition*, volume 41, 2008.

[148] X. Zhao, H. Ning, Y. Liu, and T. Huang. Discriminative estimation of 3d human pose using gaussian processes. In *19th International Conference on Pattern Recognition*, 2008.

[149] H. Zhou and H. Hu. Human motion tracking for rehabilitation–a survey. In *Biomedical Signal Processing and Control*, volume 3, 2008.